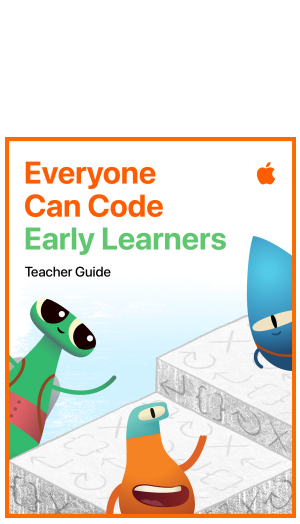
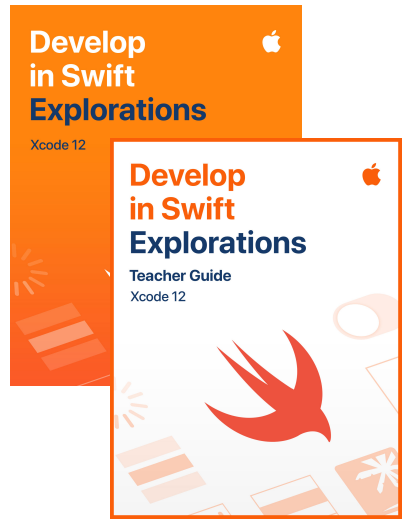
Everyone Can Code and Develop in Swift

Ontario 1–12 Correlations

Ontario Grades 1–8 Mathematics: Algebra (2020)
Ontario Grade 9 Mathematics: Algebra (2021)
Ontario Grades 10–12: Computer Studies (2008)





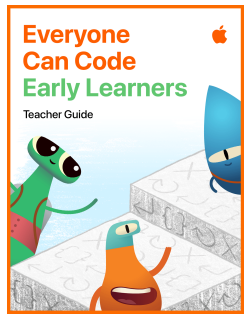
Resources

Visit Website: 2020 Ontario 1–8 Mathematics Curriulum | Coding >

Visit Website: 2021 Ontario 9 Mathematics Curriculum | Coding >

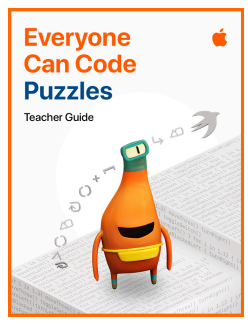
Visit Website: 2008 Ontario 10-12 Computer Studies Curriculum >

Everyone Can Code Early Learners



Download Teacher Guide >

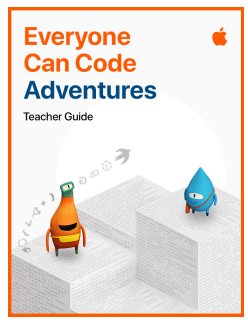
Everyone Can Code Playgrounds



Download Student Book >

Download Teacher Guide >

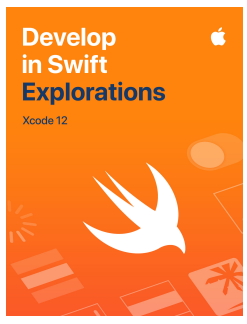
Everyone Can Code Adventures

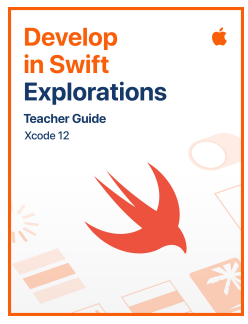


Download Student Book >

Download Teacher Guide >

Develop in Swift Explorations





Download Student Book >

Download Teacher Guide >

Contents

Ontario Correlations

Grade 1

< Back to Contents
< Back to Resources

Grade 1

Mathematics

Everyone Can Code
Early Learners Teacher Guide

Ontario Curriculum Expectations	**Commands** pp. 8–14	**Functions** pp. 15–21	**For Loops** pp. 22–28	**Variables** pp. 29–35	**App. Design** pp. 36–38
Mathematics: Algebra					
Overall Expectations					
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills	•	•	•	•	•
Specific Expectations					
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential events	•				
C3.2 Read and alter existing code, including code that involves sequential events, and describe how changes to the code affect the outcomes	•				

Grade 2

< Back to Contents
< Back to Resources

Grade 2

Mathematics

Everyone Can Code
Early Learners Teacher Guide

Ontario Curriculum Expectations	**Commands** pp. 8–14	**Functions** pp. 15–21	**For Loops** pp. 22–28	**Variables** pp. 29–35	**App Design** pp. 36–38
Mathematics: Algebra					
Overall Expectations					
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills	•	•	•	•	•
Specific Expectations					
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential and concurrent events	•	•			
C3.2 Read and alter existing code, including code that involves sequential and concurrent events, and describe how changes to the code affect the outcomes	•	•			

Grade 3

< Back to Contents
< Back to Resources

Grade 3

Mathematics

Everyone Can Code
Early Learners Teacher Guide

Ontario Curriculum Expectations	**Commands** pp. 8–14	**Functions** pp. 15–21	**For Loops** pp. 22–28	**Variables** pp. 29–35	**App Design** pp. 36–38
Mathematics: Algebra					
Overall Expectations					
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills	•	•	•	•	•
Specific Expectations					
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, and repeating events	•	•	•	•	
C3.2 Read and alter existing code, including code that involves sequential, concurrent, and repeating events, and describe how changes to the code affect the outcomes	•	•	•	•	

Grade 4

< Back to Contents
< Back to Resources

Grade 4

Mathematics

Everyone Can Code
Early Learners Teacher Guide

Ontario Curriculum Expectations	Commands pp. 8–14	Functions pp. 15–21	For Loops pp. 22–28	Variables pp. 29–35	App Design pp. 36–38
Mathematics: Algebra					
Overall Expectations					
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills	•	•	•	•	•
Specific Expectations					
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, repeating, and nested events	•	•	•	•	•
C3.2 Read and alter existing code, including code that involves sequential, concurrent, repeating, and nested events, and describe how changes to the code affect the outcomes	•	•	•	•	•

Grade 4

Mathematics

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
Mathematics: Algebra											
Overall Expectations											
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•	•	•	•
Specific Expectations											
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, repeating, and nested events		•	•	•							
C3.2 Read and alter existing code, including code that involves sequential, concurrent, repeating, and nested events, and describe how changes to the code affect the outcomes		•	•	•							

Grade 5

< Back to Contents
< Back to Resources

Grade 5

Mathematics

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
Mathematics: Algebra											
Overall Expectations											
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•	•	•	•
Specific Expectations											
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves conditional statements and other control structures		•	•	•	•	•	•	•	•	•	•
C3.2 Read and alter existing code, including code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes		•	•	•	•	•	•	•	•	•	•

Grade 6

< Back to Contents
< Back to Resources

Grade 6

Mathematics

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
Mathematics: Algebra											
Overall Expectations											
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•	•	•	•
Specific Expectations											
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing efficient code, including code that involves conditional statements and other control structures		•	•	•	•	•	•	•	•	•	•
C3.2 Read and alter existing code, including code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code		•	•	•	•	•	•	•	•	•	•

Grade 7

< Back to Contents
< Back to Resources

Grade 7

Mathematics

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
Mathematics: Algebra											
Overall Expectations											
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•	•	•	•
Specific Expectations											
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing efficient code, including code that involves events influenced by a defined count and/or sub-program and other control structures		•	•	•	•	•	•	•	•	•	•
C3.2 Read and alter existing code, including code that involves events influenced by a defined count and/or sub-program and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code		•	•	•	•	•	•	•			

Grade 7

Mathematics

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Mathematics: Algebra								
Overall Expectations								
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•
Specific Expectations								
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing efficient code, including code that involves events influenced by a defined count and/or sub-program and other control structures		•	•	•	•	•	•	•
C3.2 Read and alter existing code, including code that involves events influenced by a defined count and/or sub-program and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code		•	•	•	•	•	•	•

Grade 8

< Back to Contents
< Back to Resources

Grade 8

Mathematics

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
Mathematics: Algebra											
Overall Expectations											
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•	•	•	•
Specific Expectations											
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves the analysis of data in order to inform and communicate decisions		•	•	•	•	•	•	•	•	•	•
C3.2 read and alter existing code involving the analysis of data in order to inform and communicate decisions, and describe how changes to the code affect the outcomes and the efficiency of the code		•	•	•	•	•	•	•	•	•	•

Grade 8

Mathematics

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Mathematics: Algebra								
Overall Expectations								
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills		•	•	•	•	•	•	•
Specific Expectations								
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves the analysis of data in order to inform and communicate decisions		•	•	•	•	•	•	•
C3.2 Read and alter existing code involving the analysis of data in order to inform and communicate decisions, and describe how changes to the code affect the outcomes and the efficiency of the code		•	•	•	•	•	•	•

Grade 8

Mathematics

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–136	**Episode 1: The TV Club** pp. 137–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Mathematics: Algebra								
Overall Expectations								
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills								
Specific Expectations								
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves the analysis of data in order to inform and communicate decisions		●	●	●	●	●	●	●
C3.2 Read and alter existing code involving the analysis of data in order to inform and communicate decisions, and describe how changes to the code affect the outcomes and the efficiency of the code		●	●	●	●	●	●	●

Grade 9

< Back to Contents
< Back to Resources

Grade 9

Mathematics

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
Mathematics: Algebra											
Overall Expectations											
C2. Apply coding skills to represent mathematical concepts and relationships dynamically, and to solve problems, in algebra and across the other strands		•	•	•	•	•	•	•	•	•	•
Specific Expectations											
C2.1 Use coding to demonstrate an understanding of algebraic concepts including variables, parameters, equations, and inequalities					•	•	•	•	•	•	•
C2.2 Create code by decomposing situations into computational steps in order to represent mathematical concepts and relationships, and to solve problems		•	•	•	•	•	•	•	•	•	•
C2.3 Read code to predict its outcome, and alter code to adjust constraints, parameters, and outcomes to represent a similar or new mathematical situation		•	•	•	•	•	•	•	•	•	•

Grade 9

Mathematics

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Math: Algebra								
Overall Expectations								
C2. Apply coding skills to represent mathematical concepts and relationships dynamically, and to solve problems, in algebra and across the other strands		•	•	•	•	•	•	•
Specific Expectations								
C2.1 Use coding to demonstrate an understanding of algebraic concepts including variables, parameters, equations, and inequalities		•	•	•	•	•	•	•
C2.2 Create code by decomposing situations into computational steps in order to represent mathematical concepts and relationships, and to solve problems		•	•	•	•	•	•	•
C2.3 Read code to predict its outcome, and alter code to adjust constraints, parameters, and outcomes to represent a similar or new mathematical situation		•	•	•	•	•	•	•

Grade 9

Mathematics

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–136	**Episode 1: The TV Club** pp. 137–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Math: Algebra								
Overall Expectations								
C2. Apply coding skills to represent mathematical concepts and relationships dynamically, and to solve problems, in algebra and across the other strands		•		•		•		•
Specific Expectations								
C2.1 Use coding to demonstrate an understanding of algebraic concepts including variables, parameters, equations, and inequalities		•		•		•		•
C2.2 Create code by decomposing situations into computational steps in order to represent mathematical concepts and relationships, and to solve problems		•		•		•		•
C2.3 Read code to predict its outcome, and alter code to adjust constraints, parameters, and outcomes to represent a similar or new mathematical situation		•		•		•		•

Grade 10

ICS2O
Intro to Computer Studies

Open

Everyone Can Code
Puzzles Teacher Guide > 29

Everyone Can Code
Adventures Teacher Guide > 36

Develop in Swift
Explorations Teacher Guide > 40

< Back to Contents
< Back to Resources

ICS2O — Intro to Computer Studies

Grade 10

Everyone Can Code
Puzzles Teachers Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
Overall Expectations											
A2. Describe the different types of software products, and assess the software needs of users	•						•				
A3. Use the basic functions of an operating system correctly		•	•	•	•	•	•	•	•	•	•
B1. Describe fundamental programming concepts and constructs	•	•	•	•	•	•	•	•	•	•	•
B2. Plan and write simple programs using fundamental programming concepts		•	•	•	•	•	•	•	•	•	•
B3. Apply basic code maintenance techniques when writing programs		•	•	•	•	•	•	•	•	•	•
C1. Describe key aspects of the impact of computers and related technologies on society	•	•	•	•	•	•	•	•	•	•	•
C3. Describe legal and ethical issues related to the use of computing devices									•		
C4. Describe postsecondary education and career prospects related to computer studies											

Everyone Can Code
Puzzles Teacher Guide ›

Everyone Can Code
Adventures Teacher Guide ›

Develop in Swift
Explorations Teacher Guide ›

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
A2. Software Products											
A2.2 Assess user computing needs and select appropriate software for different situations (e.g., a student on a fixed budget, a home business user, a gaming enthusiast, a photographer, a home video enthusiast, a distance education user, a human resources manager, an accountant)	●						●				
A3. Operating Systems											
A3.3 Use general keyboard shortcuts to perform common tasks (e.g., cut, copy, paste, print, print window, print screen)		●	●	●	●	●	●	●	●	●	●
A4. Operating Systems											
A4.2 Describe the features and functions of wired and wireless networking hardware (e.g., NICs, routers, hubs, cables, modems)											
B1. Programming Concepts											
B1.1 Use correct terminology to describe programming concepts	●	●	●	●	●	●	●	●	●	●	●
B1.2 Describe the types of data that computers can process and store (e.g., numbers, text)	●	●	●	●	●	●	●	●	●	●	●
B1.3 Explain the difference between constants and variables used in programming		●			●		●	●	●	●	●

Everyone Can Code
Puzzles Teacher Guide >

Everyone Can Code
Adventures Teacher Guide >

Develop in Swift
Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
B1. Programming Concepts											
B1.5 Identify situations in which decision and looping structures are required			•	•	•	•	•	•	•	•	•
B1.6 Describe the function of Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), and arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and use them correctly in programming		•			•	•	•	•	•	•	
B2. Writing Programs											
B2.1 Use a visual problem solving model (e.g., IPO [Input, Process, Output] chart; HIPO [Hierarchy plus Input, Process, Output] chart and diagram; flow chart; storyboard) to plan the content of a program		•	•	•	•	•	•	•	•	•	•
B2.2 Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program (e.g., in a quiz program, in a unit conversion program)				•	•	•	•	•	•	•	•
B2.3 Write keyboard input and screen output statements that conform to program specifications		•	•	•	•	•	•	•	•	•	•
B2.4 Write a program that includes a decision structure for two or more choices (e.g., guessing game, rock, paper, scissors game, multiple-choice quiz, trivia game)						•	•	•	•	•	•

Everyone Can Code
Puzzles Teacher Guide >

Everyone Can Code
Adventures Teacher Guide >

Develop in Swift
Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
B2. Writing Programs											
B2.5 Write programs that use looping structures effectively (e.g., simple animation, simple board games, coin toss)				●	●	●	●	●	●	●	●
B2.6 Explain the difference between syntax, logic, and run-time errors		●	●	●	●	●	●	●	●	●	●
B2.7 Compare and contrast the use of different programming environments to solve the same problem (e.g., a solution developed in a programming language versus one developed using a spreadsheet)		●	●	●	●	●	●	●	●	●	●
B3. Computer Maintenance											
B3.1 Write clear and maintainable code using proper programming standards (e.g., indentation; naming conventions for constants, variables, and expressions)		●	●	●	●	●	●	●	●	●	●
B3.3 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in a computer program		●	●	●	●	●	●	●	●	●	●

Everyone Can Code
Puzzles Teacher Guide ›

Everyone Can Code
Adventures Teacher Guide ›

Develop in Swift
Explorations Teacher Guide ›

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
C1. Social Impact											
C1.1 Describe a variety of adaptive technologies that help to improve computer accessibility (e.g., text to speech, speech to text, adapted mouse, font control, ergonomic keyboard, virtual keyboard, sticky keys, colour contrast, image magnifier)	●	●					●				●
C1.2 Explain the impact on privacy of techniques for collecting and processing data (e.g., camera phones, reward programs, targeted advertising, digital rights management, monitoring software)									●		
C1.3 Describe how portable computing devices (e.g., PDA, cell phone, GPS, laptop) affect our everyday lives											
C1.4 Describe how electronic access to information (e.g., instant messaging, webcasts, social networking sites, wikis, blogs, video sharing sites) influences our everyday lives, as well as the lives of people in various countries around the world, in both positive and negative ways	●	●	●	●	●	●	●	●	●	●	●
C1.5 Describe issues associated with access to online services (e.g., reliability of passwords, network security, identity theft, the permanence of information released onto the Internet)									●		

Everyone Can Code
Puzzles Teacher Guide >

Everyone Can Code
Adventures Teacher Guide >

Develop in Swift
Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
C3. Ethical Issues											
C3.1 Describe legal and ethical issues related to the use of computers (e.g., music and video file downloading, spyware, identity theft, phishing, keystroke logging, packet sniffing, cyberbullying)	●	●					●				●
C3.2 Describe safeguards (e.g., effective passwords, secure websites, firewalls, biometric data) for pre– venting the unethical use of computers									●		
C4. Postsecondary Opportunities											
C4.1 Research and describe trends in careers that require computer skills, using local and national sources (e.g., local newspaper, national newspaper, career websites)	●										
C4.2 Research and report on postsecondary educational programs leading to careers in the field of information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs)	●										

Everyone Can Code
Puzzles Teacher Guide >

Everyone Can Code
Adventures Teacher Guide >

Develop in Swift
Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–12	**Functions** pp. 13–23	**For Loops** pp. 24–33	**Variables** pp. 34–46	**Conditional Code** pp. 47–59	**Types and Initialization** pp. 60–68	**Functions with Parameters** pp. 69–78	**Logical Operators** pp. 79–87	**While Loops** pp. 88–96	**Arrays and Refactoring** pp. 97–108
Computer Studies											
C4. Postsecondary Opportunities											
C4.3 Identify groups and programs that are available to support students who are interest–ed in pursuing nontraditional career choices in computer related fields (e.g., mentoring programs, virtual networking/ support groups, specialized postsecondary programs, relevant trade/industry associations)	●										
C4.4 Identify the Essential Skills and work habits that are important for success in computer studies, as defined in the Ontario Skills Passport	●										

Grade 10

ICS2O — Intro to Computer Studies

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Computer Studies								
Overall Expectations								
A2. Describe the different types of software products, and assess the software needs of users		●	●	●	●	●	●	●
A3. Use the basic functions of an operating system correctly		●	●	●	●	●	●	●
B1. Describe fundamental programming concepts and constructs		●	●	●	●	●	●	●
B2. Plan and write simple programs using fundamental programming concepts		●	●	●	●	●	●	●
B3. Apply basic code maintenance techniques when writing programs		●	●	●	●	●	●	●
C1. Describe key aspects of the impact of computers and related technologies on society							●	
C3. Describe legal and ethical issues related to the use of computing devices;							●	
C4. Describe postsecondary education and career prospects related to computer studies		●				●		●
A2. Software Products								
A2.2 Assess user computing needs and select appropriate software for different situations (e.g., a student on a fixed budget, a home business user, a gaming enthusiast, a photographer, a home video enthusiast, a distance education user, a human resources manager, an accountant)			●	●	●		●	●
A3. Operating Systems								
A3.3 Use general keyboard shortcuts to perform common tasks (e.g., cut, copy, paste, print, print window, print screen)		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Computer Studies								
A4 Home Networking								
A4.2 Describe the features and functions of wired and wireless networking hardware (e.g., NICs, routers, hubs, cables, modems)								
B1. Programming Concepts								
B1.1 Use correct terminology to describe programming concepts	●	●	●	●	●	●	●	●
B1.2 Describe the types of data that computers can process and store (e.g., numbers, text)	●	●	●	●	●	●	●	●
B1.3 Explain the difference between constants and variables used in programming		●	●	●	●	●	●	●
B1.4 Determine the expressions and instructions to use in a programming statement, taking into account the order of operations (e.g., precedence of arithmetic operators, assignment operators, and relational operators)		●	●	●	●	●	●	●
B1.5 Identify situations in which decision and looping structures are required		●	●	●	●	●	●	●
B1.6 Describe the function of Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), and arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and use them correctly in programming				●				●
B2. Writing Programs								
B2.1 Use a visual problem solving model (e.g., IPO [Input, Process, Output] chart; HIPO [Hierarchy plus Input, Process, Output] chart and diagram; flow chart; storyboard) to plan the content of a program;		●	●	●	●	●	●	●
B2.2 Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program (e.g., in a quiz program, in a unit conversion program)		●	●	●	●	●	●	●
B2.3 Write keyboard input and screen output statements that conform to program specifications		●	●	●	●		●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Computer Studies								
B2. Writing Programs								
B2.4 Write a program that includes a decision structure for two or more choices (e.g., guessing game, rock, paper, scissors game, multiple-choice quiz, trivia game)		●	●	●	●	●	●	●
B2.5 Write programs that use looping structures effectively (e.g., simple animation, simple board games, coin toss)		●	●	●	●	●	●	●
B2.6 Explain the difference between syntax, logic, and run-time errors		●	●	●	●	●	●	●
B2.7 Compare and contrast the use of different programming environments to solve the same problem (e.g., a solution developed in a programming language versus one developed using a spreadsheet)		●	●	●	●	●	●	●
B3. Computer Maintenance								
B3.1 Write clear and maintainable code using proper programming standards (e.g., indentation; naming conventions for constants, variables, and expressions)		●	●	●	●	●	●	●
B3.3 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in a computer program		●	●	●	●	●	●	●
C1. Social Impact								
C1.1 Describe a variety of adaptive technologies that help to improve computer accessibility (e.g., text to speech, speech to text, adapted mouse, font control, ergonomic keyboard, virtual keyboard, sticky keys, colour contrast, image magnifier)	●		●	●		●		
C1.2 Explain the impact on privacy of techniques for collecting and processing data (e.g., camera phones, reward programs, targeted advertising, digital rights management, monitoring software)						●	●	
C1.3 Describe how portable computing devices (e.g., PDA, cell phone, GPS, laptop) affect our everyday lives						●	●	
C1.4 Describe how electronic access to information (e.g., instant messaging, webcasts, social networking sites, wikis, blogs, video sharing sites) influences our everyday lives, as well as the lives of people in various countries around the world, in both positive and negative ways	●						●	

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Computer Studies								
C1. Social Impact								
C1.5 Describe issues associated with access to online services (e.g., reliability of passwords, network security, identity theft, the permanence of information released onto the Internet)								
C3. Ethical Issues								
C3.1 Describe legal and ethical issues related to the use of computers (e.g., music and video file downloading, spyware, identity theft, phishing, keystroke logging, packet sniffing, cyberbullying)							●	
C3.2 Describe safeguards (e.g., effective passwords, secure websites, firewalls, biometric data) for pre– venting the unethical use of computers						●		
C4. Postsecondary Opportunities								
C4.1 Research and describe trends in careers that require computer skills, using local and national sources (e.g., local newspaper, national newspaper, career websites)		●				●		●
C4.2 Research and report on postsecondary educational programs leading to careers in the field of information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs)								
C4.3 Identify groups and programs that are available to support students who are interest ed in pursuing non-traditional career choices in computer-related fields (e.g., mentoring programs, virtual networking/support groups, specialized postsecondary programs, relevant trade/industry associations)	●							
C4.4 Identify the Essential Skills and work habits that are important for success in computer studies, as defined in the Ontario Skills Passport	●							

Grade 10

ICS2O – Intro to Computer Studies

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 137–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Computer Studies								
Overall Expectations								
A2. Describe the different types of software products, and assess the software needs of users	●	●		●		●	●	●
B1. Describe fundamental programming concepts and constructs	●	●	●	●	●	●	●	●
B2. Plan and write simple programs using fundamental programming concepts		●	●	●	●	●	●	●
B3. Apply basic code maintenance techniques when writing programs		●		●		●		●
C1. Describe key aspects of the impact of computers and related technologies on society		●						
C3. Describe legal and ethical issues related to the use of computing devices;		●						
C4. Describe postsecondary education and career prospects related to computer studies	●							
A2. Software Products								
A2.2 Assess user computing needs and select appropriate software for different situations (e.g., a student on a fixed budget, a home business user, a gaming enthusiast, a photographer, a home video enthusiast, a distance education user, a human resources manager, an accountant)	●	●		●		●	●	●
A3. Operating Systems								
A3.3 Use general keyboard shortcuts to perform common tasks (e.g., cut, copy, paste, print, print window, print screen)		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Computer Studies								
A4 Home Networking								
A4.2 Describe the features and functions of wired and wireless networking hardware (e.g., NICs, routers, hubs, cables, modems)		●	●		●			●
B1. Programming Concepts								
B1.1 Use correct terminology to describe programming concepts	●	●	●	●	●	●	●	●
B1.2 Describe the types of data that computers can process and store (e.g., numbers, text)		●	●	●	●	●	●	●
B1.3 Explain the difference between constants and variables used in programming		●		●		●		●
B1.4 Determine the expressions and instructions to use in a programming statement, taking into account the order of operations (e.g., precedence of arithmetic operators, assignment operators, and relational operators)		●		●		●		●
B1.5 Identify situations in which decision and looping structures are required				●		●		●
B1.6 Describe the function of Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), and arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and use them correctly in programming		●		●		●		
B2. Writing Programs								
B2.1 Use a visual problem-solving model (e.g., IPO [Input, Process, Output] chart; HIPO [Hierarchy plus Input, Process, Output] chart and diagram; flow chart; storyboard) to plan the content of a program;		●		●		●		●
B2.2 Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program (e.g., in a quiz program, in a unit conversion program)		●		●		●		●
B2.3 Write keyboard input and screen output statements that conform to program specifications		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Computer Studies								
B2. Writing Programs								
B2.4 Write a program that includes a decision structure for two or more choices (e.g., guessing game, rock, paper, scissors game, multiple-choice quiz, trivia game)		●		●		●		●
B2.5 Write programs that use looping structures effectively (e.g., simple animation, simple board games, coin toss)				●		●		●
B2.6 Explain the difference between syntax, logic, and run-time errors		●		●		●		●
B2.7 Compare and contrast the use of different programming environments to solve the same problem (e.g., a solution developed in a programming language versus one developed using a spreadsheet)		●		●		●		●
B3. Computer Maintenance								
B3.1 Write clear and maintainable code using proper programming standards (e.g., indentation; naming conventions for constants, variables, and expressions)		●		●		●		●
B3.2 Write clear and maintainable internal documentation to a specific set of standards (e.g., program header: author, revision date, program name, program description; table of variable names and descriptions)		●		●		●		●
B3.3 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in a computer program		●		●		●		●
C1. Social Impact								
C1.1 Describe a variety of adaptive technologies that help to improve computer accessibility (e.g., text to speech, speech to text, adapted mouse, font control, ergonomic keyboard, virtual keyboard, sticky keys, colour contrast, image magnifier)		●						
C1.2 Explain the impact on privacy of techniques for collecting and processing data (e.g., camera phones, reward programs, targeted advertising, digital rights management, monitoring software)								

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Computer Studies								
C1. Social Impact								
C1.3 Describe how portable computing devices (e.g., PDA, cell phone, GPS, laptop) affect our everyday lives								
C1.4 Describe how electronic access to information (e.g., instant messaging, webcasts, social networking sites, wikis, blogs, video sharing sites) influences our everyday lives, as well as the lives of people in various countries around the world, in both positive and negative ways								
C1.5 Describe issues associated with access to online services (e.g., reliability of passwords, network security, identity theft, the permanence of information released onto the Internet)		•						
C3. Ethical Issues								
C3.1 Describe legal and ethical issues related to the use of computers (e.g., music and video file downloading, spyware, identity theft, phishing, keystroke logging, packet sniffing, cyberbullying)								
C3.2 Describe safeguards (e.g., effective passwords, secure websites, firewalls, biometric data) for preventing the unethical use of computers		•						
C4. Postsecondary Opportunities								
C4.1 Research and describe trends in careers that require computer skills, using local and national sources (e.g., local newspaper, national newspaper, career websites)								
C4.2 Research and report on postsecondary educational programs leading to careers in the field of information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs)	•							

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Computer Studies								
C4. Postsecondary Opportunities								
C4.3 Identify groups and programs that are available to support students who are interested in pursuing nontraditional career choices in computer-related fields (e.g., mentoring programs, virtual networking/ support groups, specialized postsecondary programs, relevant trade/industry associations)	•							
C4.4 Identify the Essential Skills and work habits that are important for success in computer studies, as defined in the Ontario Skills Passport								

Grade 11

ICS3C
Intro to Computer Programming

College Preparation

ICS3U
Intro to Computer Science

University Preparation

< Back to Contents
< Back to Resources

Grade 11

ICS3C — Intro to Computer Programming

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
Overall Expectations											
A1. Demonstrate the ability to use different data types, in expressions in simple computer programs		●	●	●	●	●	●	●	●	●	●
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs		●	●	●	●	●	●	●	●	●	●
A3. Use proper code maintenance techniques and conventions when creating computer programs		●	●	●	●	●	●	●	●	●	●
A1. Data Types and Expressions											
A1.1 Use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs		●	●	●	●	●	●	●	●	●	●
A1.2 Demonstrate the ability to manipulate string data in a computer program (e.g., swap two characters, capitalize first letter, extract a portion of an address, count the occurrences of a word or letter)					●		●				
A1.3 Use assignment statements correctly with both arithmetic and string expressions in computer programs (e.g., numStudents = 4 + 2, name = “Devi”)				●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A1. Data Types and Expressions											
A1.4 Use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly						●	●	●	●	●	
A2. Control Structures and Simple Algorithms											
A2.1 Write programs that incorporate user input, processing, and screen output		●	●	●	●	●	●	●	●	●	●
A2.2 Use sequence, selection, and repetition control structures to create programming solutions		●	●	●	●	●	●	●	●	●	●
A2.3 Demonstrate the ability to write algorithms with nested structures			●			●		●		●	●
A3. Code Maintenance											
A3.1 Explain the difference between syntax, logic, and run-time errors in computer programs	●	●	●	●	●	●	●	●	●	●	●
A3.2 Demonstrate the ability to correct syntax, logic, and run-time errors in computer programs		●	●	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A1. Data Types and Expressions											
A3.3 Use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation		●	●	●	●	●	●	●	●	●	●
A3.4 Demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing)		●	●	●	●	●	●	●	●	●	●
B. Software Development											
Overall Expectations											
B1. Use a variety of problem solving strategies to solve different types of problems		●	●	●	●	●	●	●	●	●	●
B2. Design software solutions to meet a variety of challenges, using a set of standards		●	●	●	●	●	●	●	●	●	●
B3. Design simple algorithms according to specifications		●	●	●	●	●	●	●	●	●	●
B1. Problem-solving Strategies											
B1.1 Use various problem-solving strategies to solve programming problems		●	●	●	●	●	●	●	●	●	●
B1.2 Use the input-process-output model to solve programming problems		●	●	●	●	●	●	●	●	●	●

ICS3C | ICS3U

Everyone Can Code Puzzles Teacher Guide > | Everyone Can Code Adventures Teacher Guide > | Develop in Swift Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B2. Designing Software Solutions											
B2.1 Design a simple program from a program template or skeleton (e.g., teacher-supplied skeleton, Help facility code snippet)		●	●	●	●	●	●	●	●	●	●
B2.2 Use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program	●	●	●	●	●	●	●	●	●	●	●
B2.3 Write subprograms (e.g., functions, procedures) that perform one well-defined task and use parameter passing and appropriate variable scope (e.g., local, global)			●					●	●		
B2.4 Use industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode) to represent the structure and components of a computer program		●	●	●	●	●	●	●	●	●	●
B3. Designing Simple Algorithms											`
B3.1 Use simple algorithms (e.g., validate entered data, count, accumulate, use random numbers) to design a program according to specifications		●	●	●	●	●	●	●	●	●	●
B3.2 Solve problems (e.g., calculation of gross pay; fuel consumption on a car trip; average of students' marks; temperature at a given altitude, using the environmental lapse rate) by applying mathematical equations or formulas in an algorithm				●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B4. The Software Development Life Cycle											
B4.2 Use a variety of techniques (e.g., dialogue, questionnaires, surveys, research) to clarify program specifications	●	●	●	●	●	●	●	●	●	●	●
B4.4 Use a test plan to test programs (i.e., identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude 'pass' or 'fail') by comparing expected to actual outcomes		●	●	●	●	●	●	●	●	●	●
B4.5 Use a variety of methods to debug programs (e.g., manual code tracing, extra code to output the state of variables)		●	●	●	●	●	●	●	●	●	●
B4.6 Communicate information about the status of a project (e.g., milestones, work completed, work outstanding) effectively in writing throughout the project		●	●	●	●	●	●	●	●	●	●
C. Computer Environments and Systems											
Overall Expectations											
C3. Use a software development environment to write and run computer programs		●	●	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
C. Computer Environments and Systems											
C3. The Software Development Environment											
C3.1 Describe the functions and features of a software development environment and use it to write and run a computer program	●	●	●	●	●	●	●	●	●	●	●
C3.3 Use Help documentation as a guide to designing and writing programs	●	●	●	●	●	●	●	●	●	●	●
D. Computers and Society											
Overall Expectations											
D3. Explain key aspects of the impact that emerging technologies have on society	●				●						
D4. Describe postsecondary education and career prospects related to computer studies	●				●						
D3. Emerging Technologies											
D3.2 Describe some emerging technologies and their implications for, and potential uses by, various members of society	●										
D3.3 Describe some of the solutions to complex problems affecting society that have been or are being developed through the use of advanced computer programming and emerging technologies	●										

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 134–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
D. Computers and Society											
D4. Post Secondary Opportunities											
D4.1 Research and describe trends in careers that require computer skills, using local and national sources (e.g., local newspaper, national newspaper, career websites)	●										
D4.3 Research and report on postsecondary educational programs leading to careers in the field of information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs).	●										
D4.5 Describe the Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport	●										

ICS3C — Intro to Computer Programming

Grade 11

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types in expressions in simple computer programs		●	●	●	●	●	●	●
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs		●	●	●	●	●	●	●
A3. Use proper code maintenance techniques and conventions when creating computer programs		●	●	●	●	●	●	●
A1. Data Types and Expressions								
A1.1 Use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs		●	●	●	●	●	●	●
A1.2 Demonstrate the ability to manipulate string data in a computer program (e.g., swap two characters, capitalize first letter, extract a portion of an address, count the occurrences of a word or letter)		●	●	●		●	●	●
A1.3 Use assignment statements correctly with both arithmetic and string expressions in computer programs (e.g., numStudents = 4 + 2, name = “Devi”)		●	●	●	●	●	●	●
A1.4 Use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly			●	●	●			●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A2. Control Structures and Simple Algorithms								
A2.1 Write programs that incorporate user input, processing, and screen output		●	●	●	●	●	●	●
A2.2 Use sequence, selection, and repetition control structures to create programming solutions		●	●	●	●	●	●	●
A2.3 Demonstrate the ability to write algorithms with nested structures				●	●	●		
A3. Code Maintenance								
A3.1 Explain the difference between syntax, logic, and run-time errors in computer programs	●	●	●	●	●	●	●	●
A3.2 Demonstrate the ability to correct syntax, logic, and run-time errors in computer programs		●	●	●	●	●	●	●
A3.3 Use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation		●	●	●	●	●	●	●
A3.4 Demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing)		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
B. Software Development								
Overall Expectations								
B1. Use a variety of problem solving strategies to solve different types of problems		●	●	●	●	●	●	●
B2. Design software solutions to meet a variety of challenges, using a set of standards		●	●	●	●	●	●	●
B3. Design simple algorithms according to specifications		●	●	●	●	●	●	●
B.1 Problem-solving Strategies								
B1.1 Use various problem-solving strategies (e.g., divide and conquer, working backwards, process analysis, examples, extreme cases, tables and charts, trial and error) to solve programming problems		●	●	●	●	●	●	●
B1.2 Use the input-process-output model to solve programming problems		●	●	●	●	●	●	●
B.2 Designing Software Solutions								
B2.1 Design a simple program from a program template or skeleton (e.g., teacher-supplied skeleton, Help facility code snippet)		●	●	●	●	●	●	●
B2.2 Use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program		●	●	●	●	●	●	●
B2.3 Write subprograms (e.g., functions, procedures) that perform one well-defined task and use parameter passing and appropriate variable scope (e.g., local, global)		●	●	●	●	●	●	●
B2.4 Use industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode) to represent the structure and components of a computer program		●	●	●	●	●	●	●
B2.5. Design user-friendly software interfaces (e.g., prompts, messages, screens, forms)								●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
B. Software Development								
B3. Designing Simple Algorithms								
B3.1 Use simple algorithms (e.g., validate entered data, count, accumulate, use random numbers) to design a program according to specifications		●	●	●	●	●	●	●
B3.2 Solve problems (e.g., calculation of gross pay; fuel consumption on a car trip; average of students' marks; temperature at a given altitude, using the environmental lapse rate) by applying mathematical equations or formulas in an algorithm		●	●	●	●	●	●	●
B4. The Software Development Life Cycle								
B4.2 Use a variety of techniques (e.g., dialogue, questionnaires, surveys, research) to clarify program specifications	●	●	●	●	●	●	●	●
B4.4 Use a test plan to test programs (i.e., identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude 'pass' or 'fail') by comparing expected to actual outcomes		●	●	●	●	●	●	●
B4.5 Use a variety of methods to debug programs (e.g., manual code tracing, extra code to output the state of variables)		●	●	●	●	●	●	●
B4.6 Communicate information about the status of a project (e.g., milestones, work completed, work outstanding) effectively in writing throughout the project		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
C. Computer Environments and Systems								
Overall Expectations								
C3. Use a software development environment to write and run computer programs		●	●	●	●	●	●	●
C3. The Software Development Environment								
C3.1 Describe the functions and features of a software development environment and use it to write and run a computer program	●	●	●	●	●	●	●	●
C3.3 Use Help documentation as a guide to designing and writing programs	●	●	●	●	●	●	●	●
D. Computers and Society								
Overall Expectations								
D2. Describe and apply procedures for safe computing to safeguard computer users and their data				●		●		
D3. Explain key aspects of the impact that emerging technologies have on society	●					●	●	
D4. Describe postsecondary education and career prospects related to computer studies	●					●		●
D2. Safe Computing								
D2.3 Describe procedures to safeguard data and programs from malware (e.g., viruses, spyware, adware)				●		●		

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
D. Software Development								
D3. Emerging Technologies								
D3.1 Explain how emerging technologies can affect personal rights and privacy (eg video surveillance, cyberbullying, identity theft)				●		●	●	
D3.2 Describe some emerging technologies and their implications for, and potential uses by, various members of society						●	●	
D3.3 Describe some of the solutions to complex problems affecting society that have been or are being developed through the use of advanced computer programming and emerging technologies (e.g., monitoring and regulating electrical supply and demand; using facial recognition programs to verify the identity of persons entering a country; analysing criminal activity by overlaying crime data on satellite imagery; analysing large-scale meteorological data to predict catastrophic storms)				●		●		
D4. Post Secondary Opportunities								
D4.1 Research and describe trends in careers that require computer skills, using local and national sources (e.g., local newspaper, national newspaper, career websites)			●			●		●
D4.3 Research and report on postsecondary educational programs leading to careers in the field of information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs)								
D4.5 Describe the Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport	●							

ICS3C — Intro to Computer Programming

Grade 11

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types in expressions in simple computer programs		●		●		●		●
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs		●		●		●		●
A3. Use proper code maintenance techniques and conventions when creating computer programs		●		●		●		●
A1. Data Types and Expressions								
A1.1 Use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs		●		●		●		●
A1.2 Demonstrate the ability to manipulate string data in a computer program (e.g., swap two characters, capitalize first letter, extract a portion of an address, count the occurrences of a word or letter)		●		●		●		●
A1.3 Use assignment statements correctly with both arithmetic and string expressions in computer programs (e.g., numStudents = 4 + 2, name = “Devi”)		●		●		●		●
A1.4 Use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly				●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A2. Control Structures and Simple Algorithms								
A2.1 Write programs that incorporate user input, processing, and screen output		●		●		●		●
A2.2 Use sequence, selection, and repetition control structures to create programming solutions		●		●		●		●
A2.3 Demonstrate the ability to write algorithms with nested structures		●		●		●		●
A3. Code Maintenance								
A3.1 Explain the difference between syntax, logic, and run-time errors in computer programs	●	●	●	●	●	●	●	●
A3.2 Demonstrate the ability to correct syntax, logic, and run-time errors in computer programs		●		●		●		●
A3.3 Use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation		●		●		●		●
A3.4 Demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing)		●		●		●		●
B.Software Development								
Overall Expectations								
B1. Use a variety of problem solving strategies to solve different types of problems		●		●		●		●
B2. Design software solutions to meet a variety of challenges, using a set of standards		●		●		●		●
B3. Design simple algorithms according to specifications		●		●		●		●
B4. Apply a software development lifecycle model to a software development project		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp.483–702
B. Software Development								
B1. Problem–solving Strategies								
B1.1 Use various problem-solving strategies (e.g., divide and conquer, working backwards, process analysis, examples, extreme cases, tables and charts, trial and error) to solve programming problems		●		●		●		●
B1.2 Use the input-process-output model to solve programming problems		●		●		●		●
B2. Designing Software Solutions								
B2.1 Design a simple program from a program template or skeleton (e.g., teacher-supplied skeleton, Help facility code snippet)		●		●		●		●
B2.2 Use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program	●	●	●	●	●	●	●	●
B2.3 Write subprograms (e.g., functions, procedures) that perform one well-defined task and use parameter passing and appropriate variable scope (e.g., local, global)		●		●		●		●
B2.4 Use industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode) to represent the structure and components of a computer program		●		●		●		●
B2.5 Design user-friendly software interfaces (e.g., prompts, messages, screens, forms)		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
B. Software Development								
B3. Designing Simple Algorithms								
B3.1 Use simple algorithms (e.g., validate entered data, count, accumulate, use random numbers) to design a program according to specifications		●		●		●		●
B3.2 Solve problems (e.g., calculation of gross pay; fuel consumption on a car trip; average of students' marks; temperature at a given altitude, using the environmental lapse rate) by applying mathematical equations or formulas in an algorithm		●		●		●		●
B4. The Software Development Life Cycle								
B4.1 Describe the phases (i.e., problem definition, analysis, design, writing code, testing, implementation, maintenance), milestones (e.g., date of completion of program specification), and products (e.g., specification, flow chart, program, documentation, bug reports) of a software development life cycle		●		●		●		●
B4.2 Use a variety of techniques (e.g., dialogue, questionnaires, surveys, research) to clarify program specifications		●		●		●		●
B4.4 Use a test plan to test programs (i.e., identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude 'pass' or 'fail') by comparing expected to actual outcomes	●	●	●	●	●	●	●	●
B4.5 Use a variety of methods to debug programs (e.g., manual code tracing, extra code to output the state of variables)		●		●		●		●
B4.6 Communicate information about the status of a project (e.g., milestones, work completed, work outstanding) effectively in writing throughout the project		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
C. Computer Environments and Systems								
Overall Expectations								
C3. Use a software development environment to write and run computer programs		●		●		●		●
C3. The Software Development Environment								
C3.1 Describe the functions and features of a software development environment and use it to write and run a computer program	●	●	●	●	●	●	●	●
C3.3 Use Help documentation as a guide to designing and writing programs	●	●	●	●	●	●	●	●
D. Computers and Society								
Overall Expectations								
D2. Describe and apply procedures for safe computing to safeguard computer users and their data			●	●	●	●		
D3. Explain key aspects of the impact that emerging technologies have on society				●	●	●		
D4. Describe postsecondary education and career prospects related to computer studies								
D2. Safe Computing								
D2.3 Describe procedures to safeguard data and programs from malware (e.g., viruses, spyware, adware)			●	●	●	●		

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
D. Computers and Society								
D3. Emerging Technologies								
D3.1 Explain how emerging technologies can affect personal rights and privacy (e.g., video surveillance, cyberbullying, identity theft)				●	●	●		
D3.2 Describe some emerging technologies and their implications for, and potential uses by, various members of society				●	●	●		
D3.3 Describe some of the solutions to complex problems affecting society that have been or are being developed through the use of advanced computer programming and emerging technologies (e.g., monitoring and regulating electrical supply and demand; using facial recognition programs to verify the identity of persons entering a country; analysing criminal activity by overlaying crime data on satellite imagery; analysing large-scale meteorological data to predict catastrophic storms)					●	●		
D4. Post Secondary Opportunities								
D4.1 Research and describe trends in careers that require computer skills, using local and national sources (e.g., local newspaper, national newspaper, career websites)	●				●	●		
D4.3 Research and report on postsecondary educational programs leading to careers in the field of information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs)	●							
D4.5 Describe the Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport								

Grade 11

ICS3U — Intro to Computer Science

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and Skills											
Overall Expectations											
A1. Demonstrate the ability to use different data types, including one-dimensional arrays, in computer programs		●	●	●	●	●	●	●	●	●	●
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs		●	●	●	●	●	●	●	●	●	●
A3. Demonstrate the ability to use subprograms within computer programs			●	●	●	●	●	●	●	●	●
A4. Use proper code maintenance techniques and conventions when creating computer programs		●	●	●	●	●	●	●	●	●	●
A1. Data Types and Expressions											
A1.1 Use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs		●	●	●	●	●	●	●	●	●	●
A1.3 Use assignment statements correctly with both arithmetic and string expressions in computer programs				●	●	●	●	●	●	●	●

ICS3C

ICS3U

Everyone Can Code
Puzzles Teacher Guide ›

Everyone Can Code
Adventures Teacher Guide

Develop in Swift
Explorations Teacher Guide ›

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A1. Data Types and Expressions											
A1.4 Demonstrate the ability to use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in computer programs						●	●	●	●	●	●
A1.5 Describe the structure of one-dimensional arrays and related concepts, including elements, indexes, and bounds											●
A1.6 Write programs that declare, initialize, modify, and access one-dimensional arrays											●
A2. Control Structures and Simple Algorithms											
A2.1 Write programs that incorporate user input, processing, and screen output		●	●	●	●	●	●	●	●	●	●
A2.2 Use sequence, selection, and repetition control structures to create programming solutions		●	●	●	●	●	●	●	●	●	●
A2.3 Write algorithms with nested structures (e.g., to count elements in an array, calculate a total, find highest or lowest value, or perform a linear search)			●			●		●		●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A3. Subprograms											
A3.1 Demonstrate the ability to use existing subprograms (e.g., random number generator, substring, absolute value) within computer programs			●	●	●	●	●	●	●	●	●
A3.2 Write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs			●	●	●	●	●	●	●	●	●
A4. Code Maintenance											
A4.1 Demonstrate the ability to identify and correct syntax, logic, and run-time errors in computer programs		●	●	●	●	●	●	●	●	●	●
A4.2 Use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation		●	●	●	●	●	●	●	●	●	●
A4.3 Demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing)		●	●	●	●	●	●	●	●	●	●
A4.4 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in computer programs		●	●	●	●	●	●	●	●	●	●
A4.5 Demonstrate the ability to validate a program using a full range of test cases		●	●	●	●	●	●	●	●	●	●

ICS3C

ICS3U

Everyone Can Code
Puzzles Teacher Guide ›

Everyone Can Code
Adventures Teacher Guide

Develop in Swift
Explorations Teacher Guide ›

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
Overall Expectations											
B1. Use a variety of problem solving strategies to solve different types of problems independently or as part of as team		●	●	●	●	●	●	●	●	●	●
B2. Design software solutions to meet a variety of challenges		●	●	●	●	●	●	●	●	●	●
B3. Design algorithms according to specifications		●	●	●	●	●	●	●	●	●	●
B4. Apply a software development lifecycle model to a software development project		●	●	●	●	●	●	●	●	●	●
B1. Problem-solving Strategies											
B1.1 Use various problem-solving strategies (e.g., stepwise refinement, divide and conquer, working backwards, examples, extreme cases, tables and charts, trial and error) when solving different types of problems		●	●	●	●	●	●	●	●	●	●
B1.2 demonstrate the ability to solve problems independently and as part of a team		●	●	●	●	●	●	●	●	●	●
B1.3 Use the input-process-output model to solve problems		●	●	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B2. Designing Software Solutions											
B2.1 Design programs from a program template or skeleton (e.g., teacher-supplied skeleton, Help facility code snippet)		●	●	●	●	●	●	●	●	●	●
B2.2 Use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program	●	●	●	●	●	●	●	●	●	●	●
B2.3 Apply the principle of modularity to design reusable code (e.g., subprograms, classes) in computer programs			●	●	●	●	●	●	●	●	●
B2.4 Represent the structure and components of a program using industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode)		●	●	●	●	●	●	●	●	●	●
B3. Designing Algorithms											
B3.1 Design simple algorithms (e.g., add data to a sorted array, delete a datum from the middle of an array) according to specifications		●	●	●	●	●	●	●	●	●	●
B3.2 Solve common problems (e.g., calculation of hypotenuse, determination of primes, calculation of area and circumference) by applying mathematical equations or formulas in an algorithm				●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B4. The Software Development Life Cycle											
B4.2 Use a variety of techniques (e.g., dialogue, questionnaires, surveys, research) to clarify program specifications;	●	●	●	●	●	●	●	●	●	●	●
B4.4 Use a test plan to test programs (i.e., identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude 'pass' or 'fail') by comparing expected to actual outcomes		●	●	●	●	●	●	●	●	●	●
B4.5 Use a variety of methods to debug programs (e.g., manual code tracing, extra code to output the state of variables)		●	●	●	●	●	●	●	●	●	●
B4.6 Communicate information about the status of a project (e.g., milestones, work completed, work outstanding) effectively in writing throughout the project		●	●	●	●	●	●	●	●	●	●
C. Computer Environments and Systems											
Overall Expectations											
C3. Use a software development environment to write and run computer programs		●	●	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
C. Computer Environments and Systems											
C3. The Software Development Environment											
C3.1 Demonstrate an understanding of an integrated software development environment and its main components (e.g., source code editor, compiler, debugger)		●	●	●	●	●	●	●	●	●	●
C3.2 Work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to design and write functioning computer programs	●	●	●	●	●	●	●	●	●	●	●
C3.3 Explain the difference between source code and machine code											
C3.4 Explain the difference between an interpreter and a compiler											
D. Topics in Computer Science											
Overall Expectations											
D2. Demonstrate an understanding of emerging areas of computer science research	●				●						

Ontario Curriculum Expectations	**Intro** pp. i–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
D. Topics in Computer Science											
D2. Exploring Computer Science											
D2.1 Demonstrate an understanding of emerging areas of research in computer science (e.g., cryptography, parallel processing, distributed computing, data mining, artificial intelligence, robotics, computer vision, image processing, human–computer interaction, security, geographic information systems [GIS])	●										
D2. Demonstrate an understanding of emerging areas of computer science research	●										

ICS3U — Intro to Computer Science

Grade 11

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types types, including one-dimensional arrays, in computer programs		●	●	●	●	●	●	●
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs		●	●	●	●	●	●	●
A3. Demonstrate the ability to use subprograms within computer programs		●	●	●	●	●	●	●
A4. Use proper code maintenance techniques and conventions when creating computer programs		●	●	●	●	●	●	●
A1. Data Types and Expressions								
A1.1 Use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs		●	●	●	●	●	●	●
A1.3 Use assignment statements correctly with both arithmetic and string expressions in computer programs		●	●	●	●	●	●	●
A1.4 Demonstrate the ability to use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in computer programs			●	●	●			●
A1.5 Describe the structure of one-dimensional arrays and related concepts, including elements, indexes, and bounds			●	●	●	●	●	●
A1.6 Write programs that declare, initialize, modify, and access one-dimensional arrays			●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A2. Control Structures and Simple Algorithms								
A2.1 Write programs that incorporate user input, processing, and screen output		●	●	●	●	●	●	●
A2.2 Use sequence, selection, and repetition control structures to create programming solutions		●	●	●	●	●	●	●
A2.3 Write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs		●	●	●	●	●	●	●
A3. Subprograms								
A3.1 Demonstrate the ability to use existing subprograms (e.g., random number generator, substring, absolute value) within computer programs		●	●	●	●	●	●	●
A3.2 Write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs		●	●	●	●	●	●	●
A4. Code Maintenance								
A4.1 Demonstrate the ability to identify and correct syntax, logic, and run-time errors in computer programs		●	●	●	●	●	●	●
A4.2 Use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation		●	●	●	●	●	●	●
A4.3 Demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing)		●	●	●	●	●	●	●
A4.4 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in computer programs		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A4. Code Maintenance								
A4.5 Demonstrate the ability to validate a program using a full range of test cases		●	●	●	●	●	●	●
B. Software Development								
Overall Expectations								
B1. Use a variety of problem solving strategies to solve different types of problems independently or as part of as team		●	●	●	●	●	●	●
B2. Design software solutions to meet a variety of challenges		●	●	●	●	●	●	●
B3. Design algorithms according to specifications		●	●	●	●	●	●	●
B4. Apply a software development life-cycle model to a software development project		●						
B1. Problem-solving Strategies								
B1.1 Use various problem-solving strategies (e.g., stepwise refinement, divide and conquer, working backwards, examples, extreme cases, tables and charts, trial and error) when solving different types of problems		●	●	●	●	●	●	●
B1.2 Demonstrate the ability to solve problems independently and as part of a team		●	●	●	●	●	●	●
B1.3 Use the input-process-output model to solve problems		●	●	●	●	●	●	●
B2. Designing Software Solutions								
B2.1 Design programs from a program template or skeleton (e.g., teacher-supplied skeleton, Help facility code snippet)		●	●	●	●	●	●	●
B2.2 Use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
B. Software Development								
B2. Designing Software Solutions								
B2.3 Apply the principle of modularity to design reusable code (e.g., subprograms, classes) in computer programs		●	●	●	●	●	●	●
B2.4 Represent the structure and components of a program using industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode)		●	●	●	●	●	●	●
B2.5 Design user-friendly software interfaces (e.g., prompts, messages, screens, forms)								●
B3. Designing Algorithms								
B3.1 Design simple algorithms (e.g., add data to a sorted array, delete a datum from the middle of an array) according to specifications			●	●	●	●	●	●
B3.2 Solve common problems (e.g., calculation of hypotenuse, determination of primes, calculation of area and circumference) by applying mathematical equations or formulas in an algorithm		●	●	●	●	●	●	●
B4. The Software Development Life Cycle								
B4.1 Describe the phases (i.e., problem definition, analysis, design, writing code, testing, implementation, maintenance), milestones (e.g., date of completion of program specification), and products (e.g., specification, flow chart, program, documentation, bug reports) of a software development life cycle		●						
B4.2 Use a variety of techniques (e.g., dialogue, questionnaires, surveys, research) to clarify program specifications	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
Software Development								
B4. The Software Development Life Cycle								
B4.4 Use a test plan to test programs (i.e., identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude 'pass' or 'fail') by comparing expected to actual outcomes		●	●	●	●	●	●	●
B4.5 Use a variety of methods to debug programs (e.g., manual code tracing, extra code to output the state of variables)		●	●	●	●	●	●	●
B4.6 Communicate information about the status of a project (e.g., milestones, work completed, work outstanding) effectively in writing throughout the project		●	●	●	●	●	●	●
C. Computer Environments and Systems								
Overall Expectations								
C3. Use a software development environment to write and run computer programs		●	●	●	●	●	●	●
C3. The Software Development Environment								
C3.1 Demonstrate an understanding of an integrated software development environment and its main components (e.g., source code editor, compiler, debugger)		●	●	●	●	●	●	●
C3.2 Work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to design and write functioning computer programs	●	●	●	●	●	●	●	●
C3.3 Explain the difference between source code and machine code								
C3.4 Explain the difference between an interpreter and a compiler								

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
D. Topics in Computer Science								
Overall Expectations								
D2. Demonstrate an understanding of emerging areas of computer science research						●		
D3. Describe postsecondary education and career prospects related to computer studies	●							
D2. Exploring Computer Science								
D2.1 Demonstrate an understanding of emerging areas of research in computer science (e.g., cryptography, parallel processing, distributed computing, data mining, artificial intelligence, robotics, computer vision, image processing, human–computer interaction, security, geographic information systems [GIS])								
D2.2 Demonstrate an understanding of an area of collaborative research between computer science and another field (e.g., bioinformatics, geology, economics, linguistics, health informatics, climatology, sociology, art)								
D2.3 Report on an area of research related to computer science, using an appropriate format (e.g., website, presentation software, video)								
D3. Postsecondary Opportunities								
D3.1 Research and describe career choices and trends in computer science, at the local, national, and international levels			●			●		●
D3.2 Identify and report on opportunities for experiential learning (e.g., co-op programs, job shadowing, career fairs) in the field of computer science								

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
D. Topics in Computer Science								
D3. Postsecondary Opportunities								
D3.3 Research and report on postsecondary educational programs leading to careers in information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs);								
D3.4 Identify groups and programs that are available to support students who are interested in pursuing non-traditional career choices related to information systems and computer science (e.g., mentoring programs, virtual networking/support groups, specialized postsecondary programs, relevant trade/industry associations)								
D3.5 Describe the Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport	●							

Grade 11

ICS3U — Intro to Computer Science

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types types, including one-dimensional arrays, in computer programs		●		●		●		●
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs		●		●		●		●
A3. Demonstrate the ability to use subprograms within computer programs		●		●		●		●
A4. Use proper code maintenance techniques and conventions when creating computer programs		●		●		●		●
A1. Data Types and Expressions								
A1.1 Use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs		●		●		●		●
A1.3 Use assignment statements correctly with both arithmetic and string expressions in computer programs		●		●		●		●
A1.4 Demonstrate the ability to use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in computer programs		●		●		●		●
A1.5 Describe the structure of one-dimensional arrays and related concepts, including elements, indexes, and bounds				●		●		●
A1.6 Write programs that declare, initialize, modify, and access one-dimensional arrays				●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A2. Control Structures and Simple Algorithms								
A2.1 Write programs that incorporate user input, processing, and screen output		●		●		●		●
A2.2 Use sequence, selection, and repetition control structures to create programming solutions		●		●		●		●
A2.3 Write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs		●		●		●		●
A3. Subprograms								
A3.1 Demonstrate the ability to use existing subprograms (e.g., random number generator, substring, absolute value) within computer programs		●		●		●		●
A3.2 Write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs		●		●		●		●
A4. Code Maintenance								
A4.1 Demonstrate the ability to identify and correct syntax, logic, and run-time errors in computer programs		●		●		●		●
A4.2 Use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation		●		●		●		●
A4.3 Demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing)		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A4. Code Maintenance								
A4.4 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in computer programs		●		●		●		●
A4.5 Demonstrate the ability to validate a program using a full range of test cases		●		●		●		●
B. Software Development								
Overall Expectations								
B1. Use a variety of problem solving strategies to solve different types of problems independently or as part of as team		●		●		●		●
B2. Design software solutions to meet a variety of challenges		●		●		●		●
B3. Design algorithms according to specifications		●		●		●		●
B4. Apply a software development lifecycle model to a software development project		●		●		●		●
B1. Problem-solving Strategies								
B1.1 Use various problem-solving strategies (e.g., stepwise refinement, divide and conquer, working backwards, examples, extreme cases, tables and charts, trial and error) when solving different types of problems		●		●		●		●
B1.2 Demonstrate the ability to solve problems independently and as part of a team		●		●		●		●
B1.3 Use the input-process-output model to solve problems		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
B. Software Development								
B2. Designing Software Solutions								
B2.1 Design programs from a program template or skeleton (e.g., teacher-supplied skeleton, Help facility code snippet)		●		●		●		●
B2.2 Use appropriate vocabulary and mode of expression (i.e., written, oral, diagrammatic) to describe alternative program designs and to explain the structure of a program	●	●	●	●	●	●	●	●
B2.3 Apply the principle of modularity to design reusable code (e.g., subprograms, classes) in computer programs		●		●		●		●
B2.4 Represent the structure and components of a program using industry-standard programming tools (e.g., structure chart, flow chart, UML [Unified Modeling Language], data flow diagram, pseudocode)		●		●		●		●
B2.5 Design user-friendly software interfaces (e.g., prompts, messages, screens, forms)		●		●		●		●
B3. Designing Algorithms								
B3.1 Design simple algorithms (e.g., add data to a sorted array, delete a datum from the middle of an array) according to specifications		●		●		●		●
B3.2 Solve common problems (e.g., calculation of hypotenuse, determination of primes, calculation of area and circumference) by applying mathematical equations or formulas in an algorithm		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
B. Software Development								
B4. The Software Development Life Cycle								
B4.1 Describe the phases (i.e., problem definition, analysis, design, writing code, testing, implementation, maintenance), milestones (e.g., date of completion of program specification), and products (e.g., specification, flow chart, program, documentation, bug reports) of a software development life cycle	●	●	●	●	●	●	●	●
B4.2 Use a variety of techniques (e.g., dialogue, questionnaires, surveys, research) to clarify program specifications		●		●		●		●
B4.4 Use a test plan to test programs (i.e., identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude 'pass' or 'fail') by comparing expected to actual outcomes		●		●		●		●
B4.5 Use a variety of methods to debug programs (e.g., manual code tracing, extra code to output the state of variables)		●		●		●		●
B4.6 Communicate information about the status of a project (e.g., milestones, work completed, work outstanding) effectively in writing throughout the project		●		●		●		●
C. Computer Environments and Systems								
Overall Expectations								
C3. Use a software development environment to write and run computer programs		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
C. Computer Environments and Systems								
C3. The Software Development Environment								
C3.1 Demonstrate an understanding of an integrated software development environment and its main components (e.g., source code editor, compiler, debugger)		●		●		●		●
C3.2 Work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to design and write functioning computer programs	●	●	●	●	●	●	●	●
C3.3 Explain the difference between source code and machine code	●							
C3.4 Explain the difference between an interpreter and a compiler						●	●	●
D. Topics in Computer Science								
Overall Expectations								
D2. Demonstrate an understanding of emerging areas of computer science research		●						
D3. Describe postsecondary education and career prospects related to computer studies	●							
D2. Exploring Computer Science								
D2.1 Demonstrate an understanding of emerging areas of research in computer science (e.g., cryptography, parallel processing, distributed computing, data mining, artificial intelligence, robotics, computer vision, image processing, human–computer interaction, security, geographic information systems [GIS])	●			●	●	●		

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
D. Topics in Computer Science								
D2. Exploring Computer Science								
D2.2 Demonstrate an understanding of an area of collaborative research between computer science and another field (e.g., bioinformatics, geology, economics, linguistics, health informatics, climatology, sociology, art)							●	●
D2.3 Report on an area of research related to computer science, using an appropriate format (e.g., website, presentation software, video)		●	●	●	●	●	●	●
D3. Postsecondary Opportunities								
D3.1 Research and describe career choices and trends in computer science, at the local, national, and international levels								
D3.2 Identify and report on opportunities for experiential learning (e.g., co-op programs, job shadowing, career fairs) in the field of computer science								
D3.3 Research and report on postsecondary educational programs leading to careers in information systems and computer science (e.g., institutions offering relevant programs, industry certifications, courses of study, entrance requirements, length of programs, costs);	●							
D3.4 Identify groups and programs that are available to support students who are interested in pursuing non-traditional career choices related to information systems and computer science (e.g., mentoring programs, virtual networking/support groups, specialized postsecondary programs, relevant trade/industry associations)	●							
D3.5 Describe the Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport								

Grade 12

ICS4C
Computer Programming

College Preparation

Everyone Can Code
Puzzles Teacher Guide > 46

Everyone Can Code
Adventures Teacher Guide > 97

Develop in Swift
Explorations Teacher Guide > 104

ICS4U
Intro to Computer Science

University Preparation

Everyone Can Code
Puzzles Teacher Guide > 111

Everyone Can Code
Adventures Teacher Guide > 73

Develop in Swift
Explorations Teacher Guide > 80

< Back to Contents
< Back to Resources

ICS4C — Computer Programming

Grade 12

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
Overall Expectations											
A1. Use data structures in the design and creation of computer programs											●
A2. Demonstrate the ability to use standard algorithms in the design and creation of computer programs		●	●	●	●	●	●	●	●	●	●
A3. Demonstrate an understanding of object-oriented programming concepts and practices in the design and creation of computer programs;											
A4. Create clear and accurate internal and external documentation to ensure the maintainability of computer software		●	●	●	●	●	●	●	●	●	●
A1. Data Structures											
A1.2 Use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in programming						●	●	●	●	●	●
A1.3 Describe the structure of one-dimensional and two-dimensional arrays and related concepts including elements, indexes, and bounds											●

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A2. Using Standard Algorithms											
A2.1 Demonstrate the ability to manipulate and convert data in a computer program (e.g., parse strings; convert data types such as numeric to string, and string to numeric; convert ‘yes’ or ‘no’ to Boolean)					●	●	●	●	●	●	●
A2.3 Demonstrate the ability to declare, initialize, modify, and access one-dimensional and two-dimensional arrays and elements within a program											●
A2.4 Demonstrate the ability to add, change, or delete elements of an array of objects in a program											●
A2.5 Demonstrate the ability to use search and sort routines (e.g., stringindexOf (“cool”), Arrayssort(intArray)) in a program											●
A3. Object-Oriented Programming											
A3.1 Explain the importance of designing reusable code in computer programs		●	●	●	●	●	●	●	●	●	●
A3.2 Explain fundamental object-oriented programming concepts (e.g., classes, objects, methods)											
A3.3 Apply the concepts of scope and visibility for variables, constants, and methods when creating classes in computer programs											
A3.4 Compare and contrast object-oriented and procedural programming paradigms											

ICS4C | ICS4U

Everyone Can Code
Puzzles Teacher Guide >

Everyone Can Code
Adventures Teacher Guide >

Develop in Swift
Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A4. Code Maintenance											
A4.1 Write maintainable computer programs by creating clear and accurate internal documentation that provides in-depth explanations of complex blocks of code		●	●	●	●	●	●	●	●	●	●
A4.2 Create clear and maintainable external user documentation (e.g., Help file, how-to manual, FAQ, installation procedures, backup and recovery procedures, training materials) as part of a complete software development project											
A4.3 Develop and implement a formal testing plan for a software development project to ensure program correctness											
B. Software Development											
Overall Expectations											
B1. Design standard algorithms according to specifications		●	●	●	●	●	●	●	●	●	●
B2. Design software solutions using object-oriented programming concepts											
B3. Design user-friendly graphical user interfaces (GUIs) that meet user requirements											
B4. Participate in a large student-managed project, using proper project management tools and techniques to manage the process effectively											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B2. Designing Software Solutions											
B1.1 Design algorithms to solve practical mathematical problems (e.g., amount of paint or carpet needed, number of shingles needed, energy costs, amount of water needed for an aquarium, projection of Aboriginal youth population growth)											
B1.2 Design algorithms that require precision and accuracy when rounding numbers (e.g., currency calculations, amortization, recipe volume changes)											
B1.3 Design data validation routines (e.g., capitalization; formatting of postal codes, telephone numbers, SINs, and alphanumeric data; length and range checking)											
B2. Object-oriented Software Solutions											
B2.1 Demonstrate the ability to create and use instance methods (e.g., constructors, mutators, accessors) in a computer program											
B2.2 Design a simple base class to represent objects or concepts in a problem statement, using program templates or skeletons											
B2.3 Write methods that require parameter passing in a computer program											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B3. Graphical User Interfaces											
B3.1 Design graphical user interfaces that contain common controls (e.g., buttons, labels, text boxes)											
B3.2 Design a user-friendly graphical user inter-face that helps to improve user accessibility (e.g., for multilingualism; for those with limited eyesight or colour vision deficiency)											
B3.3 Evaluate a user interface for conformity with a given accessibility standard (e.g., Section 508 of the Rehabilitation Act (US), W3C User Interface Domain, or a student or teacher-created standard)											
B3.4 Design responses to user events in a graphical user interface											
B4. Student-managed Project											
B4.1 Describe the phases of a model (e.g., waterfall, iterative, XP [Extreme Programming], RAD [Rapid Application Development]) of the software development life cycle											
B4.2 Create a project plan for a software development project, outlining the tasks at each phase of the software development life cycle											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
B4. Student-managed Project											
B4.3 Use project management tools (e.g., Gantt chart, PERT chart) and time management tools (e.g., organizer, calendar) to help develop a software project											
B4.4 Use industry-standard programming tools (e.g., UML [Unified Modeling Language], diagrams, structure charts, flow charts, pseudocode) to develop a software project		●	●	●	●	●	●	●	●	●	●
C. Programming Environment											
Overall Expectations											
C1. demonstrate the ability to use project management tools to plan and track activities for a software development project		●	●	●	●	●	●	●	●	●	●
C2. demonstrate the ability to use software development tools to design and write a computer program		●	●	●	●	●	●	●	●	●	●
C1. Project Management Tools											
C1.1 Use software tools (e.g., email, wikis, blogs, task lists, bulletin boards, spreadsheets, shared calendars) to plan and track activities during a software development project		●	●	●	●	●	●	●	●	●	●
C1.2 Communicate information about project status effectively in writing throughout the project		●	●	●	●	●	●	●	●	●	●

ICS4C ICS4U

Everyone Can Code Puzzles Teacher Guide > Everyone Can Code Adventures Teacher Guide > Develop in Swift Explorations Teacher Guide >

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
C. Programming Environment											
C2. Software Development Tools											
C2.1 Use the features of a software development environment to debug programs and create functioning computer programs		●	●	●	●	●	●	●	●	●	●
C2.2 Work independently, using the Help function, to resolve syntax issues while programming		●	●	●	●	●	●	●	●	●	●
C2.3 Work independently, using reference materials (e.g., code snippets, sample programs, APIs, tutorials), to design and write functioning computer programs		●	●	●	●	●	●	●	●	●	●
D. Computers and Society											
Overall Expectations											
D2. Demonstrate an understanding of ethical issues and practices related to the use of computers											
D3. Investigate and report on emerging computer technologies and their potential impact on society and the economy	●				●						
D4. Research and report on the range of career paths and lifelong learning opportunities in software development or a computer-related field	●				●						

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
D. Computers and Society											
D2. Ethical Practices											
D2.1 Investigate and describe an ethical issue related to the use of computers (e.g., piracy, privacy, security, phishing, spyware, cyberbullying).									●		
D2.2 Describe the essential elements of a code of ethics for computer programmers, and explain why there is a need for such a code (e.g., plagiarism, backdoors, spyware, unethical programming practices).											
D2.3 Outline and apply strategies to encourage ethical computing practices at home, at school, and at work.											
D3. Emerging Technologies											
D3.1 Describe the evolution of some emerging programming languages.	●	●	●	●	●	●	●	●	●	●	●
D3.2 Investigate and report on innovations in information technology (e.g., webcasting, VoIP, multiplayer online gaming) and their potential impact on society and the economy.	●										
D3.3 Describe programming requirements for a variety of emerging technologies (e.g., web programming, smartphones, embedded systems).	●										

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp. 70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
D. Computers and Society											
D4. Computer-related Careers											
D4.1 Research and report on the range of career opportunities in software development, including duties, responsibilities, qualifications, and compensation	●										
D4.2 Research and report on opportunities for lifelong learning in software development or a computer-related field	●										
D4.3 Evaluate their own development of Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport	●										

ICS4C — Computer Programming

Grade 12

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
Overall Expectations								
A1. Use data structures in the design and creation of computer programs			●	●	●	●	●	●
A2. Demonstrate the ability to use standard algorithms in the design and creation of computer programs		●	●	●	●	●	●	●
A3. Demonstrate an understanding of object-oriented programming concepts and practices in the design and creation of computer programs;		●	●	●	●	●	●	●
A4. Create clear and accurate internal and external documentation to ensure the maintainability of computer software		●	●	●	●	●	●	●
A1. Data Structures								
A1.1 Perform operations on data types typically used in business applications (e.g., express money amounts as dollars and cents, express dates and times in various formats)		●	●	●	●	●	●	●
A1.2 Use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in programming			●	●	●			●
A1.3 Describe the structure of one-dimensional and two-dimensional arrays and related concepts including elements, indexes, and bounds			●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A2. Using Standard Algorithms								
A2.1 Demonstrate the ability to manipulate and convert data in a computer program (e.g., parse strings; convert data types such as numeric to string, and string to numeric; convert ‘yes’ or ‘no’ to Boolean)		●	●	●	●	●	●	●
A2.3 Demonstrate the ability to declare, initialize, modify, and access one-dimensional and two-dimensional arrays and elements within a program			●	●	●	●	●	●
A2.4 Demonstrate the ability to add, change, or delete elements of an array of objects in a program			●	●	●	●	●	●
A2.5 Demonstrate the ability to use search and sort routines (e.g., stringindexOf (“cool”), Arrayssort(intArray)) in a program				●	●			
A3. Object-oriented Programming								
A3.1 Explain the importance of designing reusable code in computer programs		●	●	●	●	●	●	●
A3.2 Explain fundamental object-oriented programming concepts (e.g., classes, objects, methods)		●	●	●	●	●	●	●
A3.3 Apply the concepts of scope and visibility for variables, constants, and methods when creating classes in computer programs								●
A3.4 Compare and contrast object-oriented and procedural programming paradigms		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A4. Code Maintenance								
A4.1 Write maintainable computer programs by creating clear and accurate internal documentation that provides in-depth explanations of complex blocks of code		●	●	●	●	●	●	●
A4.2 Create clear and maintainable external user documentation (e.g., Help file, how-to manual, FAQ, installation procedures, backup and recovery procedures, training materials) as part of a complete software development project		●	●	●	●	●	●	●
A4.3 Develop and implement a formal testing plan for a software development project to ensure program correctness		●	●	●	●	●	●	●
B. Software Development								
Overall Expectations								
B1. Design standard algorithms according to specifications		●	●	●	●	●	●	●
B2. Design software solutions using object-oriented programming concepts		●	●	●	●	●	●	●
B2. Design software solutions using object-oriented programming concepts								●
B4. Participate in a large student-managed project, using proper project management tools and techniques to manage the process effectively								●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
B. Software Development								
B1. Designing Software Solutions								
B1.1 Design algorithms to solve practical mathematical problems (e.g., amount of paint or carpet needed, number of shingles needed, energy costs, amount of water needed for an aquarium, projection of Aboriginal youth population growth)		●	●	●	●	●	●	●
B1.2 Design algorithms that require precision and accuracy when rounding numbers (e.g., currency calculations, amortization, recipe volume changes)		●						
B1.3 Design data validation routines (e.g., capitalization; formatting of postal codes, telephone numbers, SINs, and alphanumeric data; length and range checking)								
B2. Object-oriented Software Solutions								
B2.1 Demonstrate the ability to create and use instance methods (e.g., constructors, mutators, accessors) in a computer program								●
B2.2 Design a simple base class to represent objects or concepts in a problem statement, using program templates or skeletons								●
B2.3 Write methods that require parameter passing in a computer program								
B3. Graphical User Interfaces								
B3.1 Design graphical user interfaces that contain common controls (e.g., buttons, labels, text boxes)								●
B3.2 Design a user-friendly graphical user inter-face that helps to improve user accessibility (e.g., for multilingualism; for those with limited eyesight or colour vision deficiency)								●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
B. Software Development								
B3. Graphical User Interfaces								
B3.3 Evaluate a user interface for conformity with a given accessibility standard (e.g., Section 508 of the Rehabilitation Act (US), W3C User Interface Domain, or a student or teacher-created standard)								●
B3.4 Design responses to user events in a graphical user interface								●
B4. Student-managed Project								
B4.1 Describe the phases of a model (e.g., waterfall, iterative, XP [Extreme Programming], RAD [Rapid Application Development]) of the software development life cycle								
B4.2 Describe the phases of a model (e.g., waterfall, iterative, XP [Extreme Programming], RAD [Rapid Application Development]) of the software development life cycle								●
B4.3 Use project management tools (e.g., Gantt chart, PERT chart) and time management tools (e.g., organizer, calendar) to help develop a software project		●	●	●	●	●	●	●
B4.4 Use industry-standard programming tools (e.g., UML [Unified Modeling Language], diagrams, structure charts, flow charts, pseudocode) to develop a software project		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
C. Programming Environment								
Overall Expectations								
C1. demonstrate the ability to use project management tools to plan and track activities for a software development project		●	●	●	●	●	●	●
C2. demonstrate the ability to use software development tools to design and write a computer program	●	●	●	●	●	●	●	●
C1. Project Management Tools								
C1.1 Use software tools (e.g., email, wikis, blogs, task lists, bulletin boards, spreadsheets, shared calendars) to plan and track activities during a software development project		●	●	●	●	●	●	●
C1.2 Communicate information about project status (e.g., completed, in progress, not started, problems encountered, recommended modification to deadlines) effectively in writing throughout the project		●	●	●	●	●	●	●
C2. Software Development Tools								
C2.1 Use the features of a software development environment to debug programs and create functioning computer programs		●	●	●	●	●	●	●
C2.2 Work independently, using the Help function, to resolve syntax issues while programming	●	●	●	●	●	●	●	●
C2.3 Work independently, using reference materials (e.g., code snippets, sample programs, APIs, tutorials), to design and write functioning computer programs	●	●	●	●	●	●	●	●
D. Computers and Society								
Overall Expectations								
D2. Demonstrate an understanding of ethical issues and practices related to the use of computers						●	●	
D3. Investigate and report on emerging computer technologies and their potential impact on society and the economy.	●					●	●	
D4. Research and report on the range of career paths and lifelong learning opportunities in software development or a computer-related field						●		●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
D. Computers and Society								
D2. Ethical Practices								
D2.1 Investigate and describe an ethical issue related to the use of computers (e.g., piracy, privacy, security, phishing, spyware, cyberbullying)						●	●	
D2.2 Describe the essential elements of a code of ethics for computer programmers, and explain why there is a need for such a code (e.g., plagiarism, backdoors, spyware, unethical programming practices)						●	●	
D2.3 Outline and apply strategies to encourage ethical computing practices at home, at school, and at work						●	●	
D3. Emerging Technologies								
D3.1 Describe the evolution of some emerging programming languages	●	●	●	●	●	●	●	●
D3.2 Investigate and report on innovations in information technology (e.g., webcasting, VoIP, multiplayer online gaming) and their potential impact on society and the economy							●	
D3.3 Describe programming requirements for a variety of emerging technologies (e.g., web programming, smartphones, embedded systems)			●			●	●	●
D4. Computer-related Careers								
D4.1 Research and report on the range of career opportunities in software development, including duties, responsibilities, qualifications, and compensation			●			●		●
D4.2 Research and report on opportunities for lifelong learning in software development or a computer-related field	●							
D4.3 Evaluate their own development of Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport	●							

ICS4C — Computer Programming

Grade 12

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types types, including one-dimensional arrays, in computer programs				●		●		●
A2. Demonstrate the ability to use standard algorithms in the design and creation of computer programs		●		●		●		●
A3. Demonstrate an understanding of object-oriented programming concepts and practices in the design and creation of computer programs;		●		●		●		●
A4. Create clear and accurate internal and external documentation to ensure the maintainability of computer software		●		●		●		●
A1. Data Structures								
A1.1 Perform operations on data types typically used in business applications (e.g., express money amounts as dollars and cents, express dates and times in various formats)		●		●		●		●
A1.2 Use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in programming				●		●		●
A1.3 Describe the structure of one-dimensional and two-dimensional arrays and related concepts including elements, indexes, and bounds						●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A2. Using Standard Algorithms								
A2.1 Demonstrate the ability to manipulate and convert data in a computer program (e.g., parse strings; convert data types such as numeric to string, and string to numeric; convert 'yes' or 'no' to Boolean)		●		●		●		●
A2.3 Demonstrate the ability to declare, initialize, modify, and access one-dimensional and two-dimensional arrays and elements within a program						●		●
A2.4 Demonstrate the ability to add, change, or delete elements of an array of objects in a program						●		●
A3. Object-oriented Programming								
A3.1 Explain the importance of designing reusable code in computer programs		●	●	●	●	●	●	●
A3.2 Explain fundamental object-oriented programming concepts (e.g., classes, objects, methods)		●		●		●		●
A3.3 Apply the concepts of scope and visibility for variables, constants, and methods when creating classes in computer programs		●		●		●		●
A3.4 Compare and contrast object-oriented and procedural programming paradigms		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A4. Code Maintenance								
A4.1 Write maintainable computer programs by creating clear and accurate internal documentation that provides in-depth explanations of complex blocks of code		●		●		●		●
A4.2 Create clear and maintainable external user documentation (e.g., Help file, how-to manual, FAQ, installation procedures, backup and recovery procedures, training materials) as part of a complete software development project		●		●		●		●
A4.3 Develop and implement a formal testing plan for a software development project to ensure program correctness		●		●		●		●
B. Software Development								
Overall Expectations								
B1. Design standard algorithms according to specifications		●		●		●		●
B2. Design software solutions using object-oriented programming concepts		●		●		●		●
B3. Design user-friendly graphical user interfaces (GUIs) that meet user requirements		●		●		●		●
B4. Participate in a large student-managed project, using proper project management tools and techniques to manage the process effectively		●		●		●		●
B2. Designing Software Solutions								
B1.1 Design algorithms to solve practical mathematical problems (e.g., amount of paint or carpet needed, number of shingles needed, energy costs, amount of water needed for an aquarium, projection of Aboriginal youth population growth)		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
B. Software Development								
B2. Designing Software Solutions								
B1.2 Design algorithms that require precision and accuracy when rounding numbers (e.g., currency calculations, amortization, recipe volume changes)								●
B1.3 Design data validation routines (e.g., capitalization; formatting of postal codes, telephone numbers, SINs, and alphanumeric data; length and range checking)								●
B2. Object-oriented Software Solutions								
B2.1 Demonstrate the ability to create and use instance methods (e.g., constructors, mutators, accessors) in a computer program						●		●
B2.2 Design a simple base class to represent objects or concepts in a problem statement, using program templates or skeletons						●		●
B2.3 Write methods that require parameter passing in a computer program						●		●
B3. Graphical User Interfaces								
B3.1 Design graphical user interfaces that contain common controls (e.g., buttons, labels, text boxes)		●		●		●		●
B3.2 Design a user-friendly graphical user inter-face that helps to improve user accessibility (e.g., for multilingualism; for those with limited eyesight or colour vision deficiency)		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
B. Software Development								
B3. Graphical User Interfaces								
B3.3 Evaluate a user interface for conformity with a given accessibility standard (e.g., Section 508 of the Rehabilitation Act (US), W3C User Interface Domain, or a student or teacher-created standard)		●		●		●		●
B3.4 Design responses to user events in a graphical user interface		●		●		●		●
B4. Student-managed Project								
B4.1 Describe the phases of a model (e.g., waterfall, iterative, XP [Extreme Programming], RAD [Rapid Application Development]) of the software development life cycle		●		●		●		●
B4.2 Create a project plan for a software development project, outlining the tasks at each phase of the software development life cycle		●		●		●		●
B4.3 Use project management tools (e.g., Gantt chart, PERT chart) and time management tools (e.g., organizer, calendar) to help develop a software project		●		●		●		●
B4.4 Use industry-standard programming tools (e.g., UML [Unified Modeling Language], diagrams, structure charts, flow charts, pseudocode) to develop a software project		●		●		●		●
C. Programming Environment								
Overall Expectations								
C1. demonstrate the ability to use project management tools to plan and track activities for a software development project		●		●		●		●
C2. demonstrate the ability to use software development tools to design and write a computer program		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
C. Programming Environment								
C1. Project Management Tools								
C1.1 Use software tools (e.g., email, wikis, blogs, task lists, bulletin boards, spreadsheets, shared calendars) to plan and track activities during a software development project		●		●		●		●
C1.2 Communicate information about project status (e.g., completed, in progress, not started, problems encountered, recommended modification to deadlines) effectively in writing throughout the project		●		●		●		●
C2. Software Development Tools								
C2.1 Use the features of a software development environment to debug programs and create functioning computer programs		●		●		●		●
C2.2 Work independently, using the Help function, to resolve syntax issues while programming		●		●		●		●
C2.3 Work independently, using reference materials (e.g., code snippets, sample programs, APIs, tutorials), to design and write functioning computer programs	●	●	●	●	●	●	●	●
D. Computers and Society								
Overall Expectations								
D2. Demonstrate an understanding of ethical issues and practices related to the use of computers	●			●	●	●	●	●
D3. Investigate and report on emerging computer technologies and their potential impact on society and the economy	●	●			●			●
D4. Research and report on the range of career paths and lifelong learning opportunities in software development or a computer-related field	●							

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp. 169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
D. Computers and Society								
D2. Ethical Practices								
D2.1 Investigate and describe an ethical issue related to the use of computers (e.g., piracy, privacy, security, phishing, spyware, cyberbullying)	●			●	●	●	●	●
D2.2 Describe the essential elements of a code of ethics for computer programmers, and explain why there is a need for such a code (e.g., plagiarism, backdoors, spyware, unethical programming practices)	●			●	●	●	●	●
D2.3 Outline and apply strategies to encourage ethical computing practices at home, at school, and at work	●			●	●	●	●	●
D3. Emerging Technologies								
D3.1 Describe the evolution of some emerging programming languages	●	●	●	●	●	●	●	●
D3.2 Investigate and report on innovations in information technology (e.g., webcasting, VoIP, multiplayer online gaming) and their potential impact on society and the economy	●	●			●			●
D3.3 Describe programming requirements for a variety of emerging technologies (e.g., web programming, smartphones, embedded systems)	●	●			●			●
D4. Computer-related Careers								
D4.1 Research and report on the range of career opportunities in software development, including duties, responsibilities, qualifications, and compensation	●							
D4.2 Research and report on opportunities for lifelong learning in software development or a computer-related field	●							
D4.3 Evaluate their own development of Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport								

ICS4U — Computer Science

Grade 12

Everyone Can Code
Puzzles Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
Overall Expectations											
A1. Demonstrate the ability to use different data types and expressions when creating computer programs						●	●	●	●		●
A2. Describe and use modular programming concepts and principles in the creation of computer programs											
A3. Design and write algorithms and subprograms to solve a variety of problems											
A4. Use proper code maintenance techniques when creating computer programs		●	●	●	●	●	●	●	●	●	●
A1. Data Types and Expressions											
A1.2 Demonstrate an understanding of type conversion (e.g., string-to-integer, character-to-integer, integer-to-character, floating point-to-integer, casting in an inheritance hierarchy)											
A1.3 Demonstrate the ability to use non-numeric comparisons (e.g., strings, comparable interface) in computer programs											
A1.4 Demonstrate an understanding of the limitations of finite data representations (e.g., integer bounds, precision of floating-point real numbers, rounding errors) when designing algorithms											●

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A1. Data Types and Expressions											
A1.5 Describe and use one-dimensional arrays of compound data types (e.g., objects, structures, records) in a computer program											●
A2. Modular Programming											
A2.1 Create a modular program that is divided among multiple files (e.g., user-defined classes, libraries, modules)											
A2.2 Use modular design concepts that support reusable code (e.g., encapsulation, inheritance, method overloading, method overriding, polymorphism)											
A2.3 Demonstrate the ability to modify existing modular program code to enhance the functionality of a program											
A3. Designing Algorithms											
A3.2 Create linear and binary search algorithms to find data in an array											●
A3.3 Create subprograms to insert and delete array elements											
A3.4 Create a sort algorithm (e.g., bubble, insertion, selection) to sort data in an array											
A3.5 Create algorithms to process elements in two-dimensional arrays (e.g., multiply each element by a constant, interchange elements, multiply matrices, process pixels in an image)											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
A. Programming Concepts and skills											
A3. Designing Algorithms											
A3.6 Design a simple and efficient recursive algorithm (e.g., calculate a factorial, translate numbers into words, perform a merge sort, generate fractals, perform XML parsing)											
A4. Code Maintenance											
A4.1 Work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to resolve syntax issues during software development		●	●	●	●	●	●	●	●	●	●
A4.2 Develop and implement a formal testing plan (e.g., unit testing, integration testing, regression testing) for a software project to ensure program correctness		●	●	●	●	●	●	●	●	●	●
A4.2 Develop and implement a formal testing plan (e.g., unit testing, integration testing, regression testing) for a software project to ensure program correctness											
A4.4 Create clear and maintainable external user documentation (e.g., Help files, training materials, user manuals)		●	●	●	●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B. Software Development											
Overall Expectations											
B1. Demonstrate the ability to manage the software development process effectively, through all of its stages – planning, development, production, and closing											
B2. Apply standard project management techniques in the context of a student-managed team project											
B1. Project Management											
B1.1 Create a software project plan by producing a software scope document and determining the tasks, deliverables, and schedule											
B1.2 Develop the software product according to the project plan (i.e., ensure that the software meets end user needs, functions as intended, and can be produced within quality standards, budget, and timelines)											
B1.3 Produce the software according to specifications (i.e., code, test, deploy), and create user documentation and training materials											
B1.4 Use an appropriate project management tool (e.g., Gantt chart, PERT chart, calendar) to manage project components											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
B.Software Development											
B1. Project Management											
B1.5 Close the project (i.e., confirm that software meets all user requirements, deliver software in appropriate format, plan software support and maintenance)											
B1.6 Review the management of the project (e.g., compare plan to actual performance, outline successes, make recommendations for improvement) and prepare a report in an appropriate format											
B1.7 Demonstrate the ability to use shared resources to manage source code effectively and securely (e.g., organize software components using shared files and folders with timestamps, and proper version control)											
B2. Software Project Contribution											
B2.1 Demonstrate the ability to contribute, as a team member, to the planning, development, and production of a large software project											
B2.2 Demonstrate the ability to meet project goals and deadlines by managing individual time during a group project											
B2.3 Reflect on, and assess, team and individual progress during the project review											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
C. Design Modular Programs											
Overall Expectations											
C1. Demonstrate the ability to apply modular design concepts in computer programs		●	●	●	●	●	●	●	●	●	●
C2. Analyse algorithms for their effectiveness in solving a problem		●	●	●	●	●	●	●	●	●	●
C1. Modular Design											
C1.1 Decompose a problem into modules, classes, or abstract data types (e.g., stack, queue, dictionary) using an object-oriented design methodology (e.g., CRC [Class Responsibility Collaborator] or UML [Unified Modeling Language])											
C1.2 Demonstrate the ability to apply data encapsulation in program design (e.g., classes, records, structures)											
C1.3 Demonstrate the ability to apply the process of functional decomposition in subprogram design.											
C1.4 Apply the principle of reusability in program design (e.g., in modules, subprograms, classes, methods, and inheritance)											

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
C. Design Modular Programs											
C2. Algorithm Analysis											
C2.1 Demonstrate the ability to analyse a precondition (i.e., starting state) and a postcondition (i.e., ending state) in an algorithm											
C2.2 Compare the efficiency of linear and binary searches, using run times and computational complexity analysis (e.g., to analyse the number of statements executed, the number of iterations of a loop, or the number of comparisons performed)											
C2.3 Compare the efficiency of sorting algorithms, using run times and computational complexity analysis (e.g., to analyse the number of statements executed, the number of iterations of a loop, or the number of comparisons performed).											
C2.4 Identify common pitfalls in recursive functions (e.g., infinite recursion, exponential growth in recursive algorithms such as Fibonacci numbers)											
D. Topics in Computer Science											
Overall Expectations											
D2. Analyse ethical issues and propose strategies to encourage ethical practices related to the use of computers	●				●						
D3. Analyse the impact of emerging computer technologies on society and the economy	●				●						

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
D. Topics in Computer Science											
Overall Expectations											
D4. Research and report on different areas of research in computer science, and careers related to computer science	●				●						
D2. Ethical Practices											
D2.1 Investigate and analyse an ethical issue related to the use of computers (e.g., sharing passwords, music and video file downloading, software piracy, keystroke logging, phishing, cyberbullying)											
D2.2 Describe the essential elements of a code of ethics for computer programmers (e.g., ACM [Association for Computing Machinery] and IEEE [Institute of Electrical and Electronics Engineers] standards) and explain why there is a need for such a code (e.g., plagiarism, backdoors, viruses, spyware, logic bombs).											
D3. Emerging Technologies and Society											
D3.1 Explain the impact of a variety of emerging technologies on various members of society and on societies and cultures around the world and on the economy					●						
D3.2 Investigate an emerging technology and produce a report using an appropriate format (e.g., technical report, website, presentation software, video)					●						

Ontario Curriculum Expectations	**Intro** pp. I–xiv	**Commands** pp. 1–24	**Functions** pp. 25–50	**For Loops** pp. 51–69	**Variables** pp.70–92	**Conditional Code** pp. 93–115	**Types and Initialization** pp. 116–134	**Functions with Parameters** pp. 135–154	**Logical Operators** pp. 155–173	**While Loops** pp. 174–192	**Arrays and Refactoring** pp. 193–214
D. Topics in Computer Science											
D4. Exploring Computer Science											
D4.1 Report on some areas of collaborative research between computer science and other fields (e.g., bioinformatics, geology, economics, linguistics, health informatics, climatology, sociology, art), on the basis of information found in industry publications (e.g., from the ACM and IEEE)											
D4.2 Investigate a topic in theoretical computer science (e.g., cryptography, graph theory, logic, computability theory, attribute grammar, automata theory, data mining, artificial intelligence, robotics, computer vision, image processing), and produce a report, using an appropriate format (e.g., website, presentation software, video).					●						
D4.3 Research and describe careers associated with computer studies (e.g., computer scientist, software engineer, systems analyst), and the postsecondary education required to prepare for them					●						
D4.4 Evaluate their own development of Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport.	●										

Grade 12

ICS4U— Computer Science

Everyone Can Code
Adventures Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types and expressions when creating computer programs		●	●	●	●	●	●	●
A2. Describe and use modular programming concepts and principles in the creation of computer programs								●
A3. Design and write algorithms and subprograms to solve a variety of problems		●	●	●	●	●	●	●
A4. Use proper code maintenance techniques when creating computer programs		●	●	●	●	●	●	●
A1. Data Types and Expressions								
A1.1 Demonstrate the ability to use integer division and resultant remainders in computer programs								
A1.2 Demonstrate an understanding of type conversion (e.g., string-to-integer, character-to- integer, integer-to-character, floating point-to-integer, casting in an inheritance hierarchy)								
A1.3 Demonstrate the ability to use non-numeric comparisons (e.g., strings, comparable interface) in computer programs								
A1.4 Demonstrate an understanding of the limitations of finite data representations (e.g., integer bounds, precision of floating-point real numbers, rounding errors) when designing algorithms								
A1.5 Describe and use one-dimensional arrays of compound data types (e.g., objects, structures, records) in a computer program			●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A2. Modular Programming								
A2.1 Create a modular program that is divided among multiple files (e.g., user-defined classes, libraries, modules)								
A2.2 Use modular design concepts that support reusable code (e.g., encapsulation, inheritance, method overloading, method overriding, polymorphism)								●
A2.3 Demonstrate the ability to modify existing modular program code to enhance the functionality of a program								●
A3. Designing Algorithms								
A3.1 Demonstrate the ability to read from, and write to, an external file (e.g., text file, binary file, data-base, XML file) from within a computer program								
A3.2 Create linear and binary search algorithms to find data in an array				●				●
A3.3 Create subprograms to insert and delete array elements			●	●		●		●
A3.4 Create a sort algorithm (e.g., bubble, insertion, selection) to sort data in an array				●				
A3.5 Create algorithms to process elements in two-dimensional arrays (e.g., multiply each element by a constant, interchange elements, multiply matrices, process pixels in an image)				●				
A3.6 Design a simple and efficient recursive algorithm (e.g., calculate a factorial, translate numbers into words, perform a merge sort, generate fractals, perform XML parsing)								

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
A. Programming Concepts and skills								
A4. Code Maintenance								
A4.1 Work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to resolve syntax issues during software development.	●	●	●	●	●	●	●	●
A4.2 Develop and implement a formal testing plan (e.g., unit testing, integration testing, regression testing) for a software project to ensure program correctness.		●	●	●	●	●	●	●
A4.3 Create fully documented program code according to industry standards (e.g., doc comments, docstrings, block comments, line comments).		●	●	●	●	●	●	●
A4.4 Create clear and maintainable external user documentation (e.g., Help files, training materials, user manuals).		●	●	●	●	●	●	●
B. Software Development								
Overall Expectations								
B1. Demonstrate the ability to manage the software development process effectively, through all of its stages — planning, development, production, and closing								
B2. Apply standard project management techniques in the context of a student-managed team project		●	●	●	●	●	●	●
B1. Project Management								
B1.1 Create a software project plan by producing a software scope document and determining the tasks, deliverables, and schedule								
B1.2 Develop the software product according to the project plan (i.e., ensure that the software meets end user needs, functions as intended, and can be produced within quality standards, budget, and timelines)								
B1.3 Produce the software according to specifications (i.e., code, test, deploy), and create user documentation and training materials								

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
B. Software Development								
B1. Project Management								
B1.4 Use an appropriate project management tool (e.g., Gantt chart, PERT chart, calendar) to manage project components		●	●	●	●	●	●	●
B1.5 Close the project (i.e., confirm that software meets all user requirements, deliver software in appropriate format, plan software support and maintenance)								
B1.6 Review the management of the project (e.g., compare plan to actual performance, outline successes, make recommendations for improvement) and prepare a report in an appropriate format								
B1.7 Demonstrate the ability to use shared resources to manage source code effectively and securely (e.g., organize software components using shared files and folders with timestamps, and proper version control)								
B2. Software Project Contribution								
B2.1 Demonstrate the ability to contribute, as a team member, to the planning, development, and production of a large software project								
B2.2 Demonstrate the ability to meet project goals and deadlines by managing individual time during a group project		●	●	●	●	●	●	●
B2.3 Reflect on, and assess, team and individual progress during the project review		●	●	●	●	●	●	●
C. Designing Modular Programs								
Overall Expectations								
C1. Demonstrate the ability to apply modular design concepts in computer programs								●
C2. Analyse algorithms for their effectiveness in solving a problem		●	●	●	●	●	●	●

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
C. Designing Modular Programs								
C1. Modular Design								
C1.1 Decompose a problem into modules, classes, or abstract data types (e.g., stack, queue, dictionary) using an object-oriented design methodology (e.g., CRC [Class Responsibility Collaborator] or UML [Unified Modeling Language])								●
C1.2 Demonstrate the ability to apply data encapsulation in program design (e.g., classes, records, structures)								●
C1.3 Demonstrate the ability to apply the process of functional decomposition in subprogram design								●
C1.4 Apply the principle of reusability in program design (e.g., in modules, subprograms, classes, methods, and inheritance)								●
C2. Algorithm Analysis								
C2.1 Demonstrate the ability to analyse a precondition (i.e., starting state) and a postcondition (i.e., ending state) in an algorithm		●	●	●	●	●	●	●
C2.2 Compare the efficiency of linear and binary searches, using run times and computational complexity analysis (e.g., to analyse the number of statements executed, the number of iterations of a loop, or the number of comparisons performed)								
C2.3 Compare the efficiency of sorting algorithms, using run times and computational complexity analysis (e.g., to analyse the number of statements executed, the number of iterations of a loop, or the number of comparisons performed)								
C2.4 Identify common pitfalls in recursive functions (e.g., infinite recursion, exponential growth in recursive algorithms such as Fibonacci numbers)								

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
D. Topics in Computer Science								
Overall Expectations								
D2. Analyse ethical issues and propose strategies to encourage ethical practices related to the use of computers				●		●	●	
D3. Analyse the impact of emerging computer technologies on society and the economy							●	
D4. Research and report on different areas of research in computer science, and careers related to computer science		●				●	●	●
D2. Ethical Practices								
D2.1 Investigate and analyse an ethical issue related to the use of computers (e.g., sharing passwords, music and video file downloading, software piracy, keystroke logging, phishing, cyberbullying)				●		●	●	
D2.2 Describe the essential elements of a code of ethics for computer programmers (e.g., ACM [Association for Computing Machinery] and IEEE [Institute of Electrical and Electronics Engineers] standards) and explain why there is a need for such a code (e.g., plagiarism, backdoors, viruses, spyware, logic bombs)				●		●	●	
D3. Emerging Technologies and Society								
D3.1 Explain the impact of a variety of emerging technologies on various members of society and on societies and cultures around the world and on the economy							●	
D3.2 Investigate an emerging technology and produce a report using an appropriate format (e.g., technical report, website, presentation software, video)							●	

Ontario Curriculum Expectations	**Intro** pp. 1–9	**Objects in Views** pp. 10–47	**Events and Handlers** pp. 48–72	**Arrays** pp. 73–104	**More Events and Handlers** pp. 105–129	**Functions and Arguments** pp. 130–160	**Return Types and Output** pp. 161–179	**Classes and Components** pp. 180–221
D. Topics in Computer Science								
D4. Exploring Computer Science								
D4.1 Report on some areas of collaborative research between computer science and other fields (e.g., bioinformatics, geology, economics, linguistics, health informatics, climatology, sociology, art), on the basis of information found in industry publications (e.g., from the ACM and IEEE)							●	
D4.2 Investigate a topic in theoretical computer science (e.g., cryptography, graph theory, logic, computability theory, attribute grammar, automata theory, data mining, artificial intelligence, robotics, computer vision, image processing), and produce a report, using an appropriate format (e.g., website, presentation software, video)							●	
D4.3 Research and describe careers associated with computer studies (e.g., computer scientist, software engineer, systems analyst), and the postsecondary education required to prepare for them		●				●	●	●
D4.4 Evaluate their own development of Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport	●							

ICS4U — Computer Science

Grade 12

Develop in Swift
Explorations Teacher Guide

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
Overall Expectations								
A1. Demonstrate the ability to use different data types and expressions when creating computer programs		●		●		●		●
A2. Describe and use modular programming concepts and principles in the creation of computer programs				●		●		●
A3. Design and write algorithms and subprograms to solve a variety of problems		●		●		●		●
A4. Use proper code maintenance techniques when creating computer programs		●		●		●		●
A1. Data Types and Expressions								
A1.1 Demonstrate the ability to use integer division and resultant remainders in computer programs								
A1.2 Demonstrate an understanding of type conversion (e.g., string-to-integer, character-to- integer, integer-to-character, floating point-to-integer, casting in an inheritance hierarchy)								
A1.3 Demonstrate the ability to use nonnumeric comparisons (e.g., strings, comparable interface) in computer programs								●
A1.4 Demonstrate an understanding of the limitations of finite data representations (e.g., integer bounds, precision of floating-point real numbers, rounding errors) when designing algorithms						●		●
A1.5 Describe and use one-dimensional arrays of compound data types (e.g., objects, structures, records) in a computer program						●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A2. Modular Programming								
A2.1 Create a modular program that is divided among multiple files (e.g., user-defined classes, libraries, modules)								
A2.2 Use modular design concepts that support reusable code (e.g., encapsulation, inheritance, method overloading, method overriding, polymorphism)				●		●		●
A2.3 Demonstrate the ability to modify existing modular program code to enhance the functionality of a program				●		●		●
A3. Designing Algorithms								
A3.1 Demonstrate the ability to read from, and write to, an external file (e.g., text file, binary file, data-base, XML file) from within a computer program								
A3.2 Create linear and binary search algorithms to find data in an array						●		
A3.3 Create subprograms to insert and delete array elements						●		●
A3.4 Create a sort algorithm (e.g., bubble, insertion, selection) to sort data in an array						●		●
A3.5 Create algorithms to process elements in two-dimensional arrays (e.g., multiply each element by a constant, interchange elements, multiply matrices, process pixels in an image)								
A3.6 Design a simple and efficient recursive algorithm (e.g., calculate a factorial, translate numbers into words, perform a merge sort, generate fractals, perform XML parsing)								

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
A. Programming Concepts and skills								
A4. Code Maintenance								
A4.1 Work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to resolve syntax issues during software development	●	●	●	●	●	●	●	●
A4.2 Develop and implement a formal testing plan (e.g., unit testing, integration testing, regression testing) for a software project to ensure program correctness		●		●		●		●
A4.3 Create fully documented program code according to industry standards (e.g., doc comments, docstrings, block comments, line comments)		●		●		●		●
A4.4 Create clear and maintainable external user documentation (e.g., Help files, training materials, user manuals)		●		●		●		●
B. Software Development								
Overall Expectations								
B1. Demonstrate the ability to manage the software development process effectively, through all of its stages — planning, development, production, and closing								●
B2. Apply standard project management techniques in the context of a student-managed team project		●		●		●		●
B1. Project Management								
B1.1 Create a software project plan by producing a software scope document and determining the tasks, deliverables, and schedule		●						●
B1.2 Develop the software product according to the project plan (i.e., ensure that the software meets end user needs, functions as intended, and can be produced within quality standards, budget, and timelines)		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
B. Software Development								
B1. Project Management								
B1.3 Produce the software according to specifications (i.e., code, test, deploy), and create user documentation and training materials		●		●		●		●
B1.4 Use an appropriate project management tool (e.g., Gantt chart, PERT chart, calendar) to manage project components		●		●		●		●
B1.5 Close the project (i.e., confirm that software meets all user requirements, deliver software in appropriate format, plan software support and maintenance)		●		●		●		●
B1.6 Review the management of the project (e.g., compare plan to actual performance, outline successes, make recommendations for improvement) and prepare a report in an appropriate format		●		●		●		●
B1.7 Demonstrate the ability to use shared resources to manage source code effectively and securely (e.g., organize software components using shared files and folders with timestamps, and proper version control)		●		●		●		●
B2. Software Project Contribution								
B2.1 Demonstrate the ability to contribute, as a team member, to the planning, development, and production of a large software project		●		●		●		●
B2.2 Demonstrate the ability to meet project goals and deadlines by managing individual time during a group project		●		●		●		●
B2.3 Reflect on, and assess, team and individual progress during the project review		●		●		●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
C. Designing Modular Programs								
Overall Expectations								
C1. Demonstrate the ability to apply modular design concepts in computer programs				●		●		●
C2. Analyse algorithms for their effectiveness in solving a problem		●		●		●		●
C1. Modular Design								
C1.1 Decompose a problem into modules, classes, or abstract data types (e.g., stack, queue, dictionary) using an object-oriented design methodology (e.g., CRC [Class Responsibility Collaborator] or UML [Unified Modeling Language])				●		●		●
C1.2 Demonstrate the ability to apply data encapsulation in program design (e.g., classes, records, structures)				●		●		●
C1.3 Demonstrate the ability to apply the process of functional decomposition in subprogram design				●		●		●
C1.4 Apply the principle of reusability in program design (e.g., in modules, subprograms, classes, methods, and inheritance)				●		●		●
C2. Algorithm Analysis								
C2.1 Demonstrate the ability to analyse a precondition (i.e., starting state) and a postcondition (i.e., ending state) in an algorithm				●		●		●
C2.2 Compare the efficiency of linear and binary searches, using run times and computational complexity analysis (e.g., to analyse the number of statements executed, the number of iterations of a loop, or the number of comparisons performed)						●		

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
C. Designing Modular Programs								
C2. Algorithm Analysis								
C2.3 Compare the efficiency of sorting algorithms, using run times and computational complexity analysis (e.g., to analyse the number of statements executed, the number of iterations of a loop, or the number of comparisons performed)						●		
C2.4 Identify common pitfalls in recursive functions (e.g., infinite recursion, exponential growth in recursive algorithms such as Fibonacci numbers)								
D. Topics in Computer Science								
Overall Expectations								
D2. Analyse ethical issues and propose strategies to encourage ethical practices related to the use of computers		●	●	●	●	●		●
D3. Analyse the impact of emerging computer technologies on society and the economy	●			●	●	●		
D4. Research and report on different areas of research in computer science, and careers related to computer science								
D2. Ethical Practices								
D2.1 investigate and analyse an ethical issue related to the use of computers (e.g., sharing passwords, music and video file downloading, software piracy, keystroke logging, phishing, cyberbullying)		●	●	●	●	●		●
D2.2 Describe the essential elements of a code of ethics for computer programmers (e.g., ACM [Association for Computing Machinery] and IEEE [Institute of Electrical and Electronics Engineers] standards) and explain why there is a need for such a code (e.g., plagiarism, backdoors, viruses, spyware, logic bombs)		●	●	●	●	●		●

Ontario Curriculum Expectations	**Intro** pp. 1–27	**Unit 1: Values** pp. 28–138	**Episode 1: The TV Club** pp. 139–168	**Unit 2: Algorithms** pp.169–273	**Episode 2: The Viewing Party** pp. 274–299	**Unit 3: Organizing Data** pp. 300–448	**Episode 3: Sharing Photos** pp. 449–482	**Unit 4: Building Apps** pp. 483–702
Topics in Computer Science								
D3. Emerging Technologies and Society								
D3.1 Explain the impact of a variety of emerging technologies on various members of society and on societies and cultures around the world and on the economy	●			●	●	●		
D3.2 Investigate an emerging technology and produce a report using an appropriate format (e.g., technical report, website, presentation software, video)	●			●	●	●		
D4. Exploring Computer Science								
D4.1 Report on some areas of collaborative research between computer science and other fields (e.g., bioinformatics, geology, economics, linguistics, health informatics, climatology, sociology, art), on the basis of information found in industry publications (e.g., from the ACM and IEEE)							●	
D4.2 Investigate a topic in theoretical computer science (e.g., cryptography, graph theory, logic, computability theory, attribute grammar, automata theory, data mining, artificial intelligence, robotics, computer vision, image processing), and produce a report, using an appropriate format (e.g., website, presentation software, video)								
D4.3 Research and describe careers associated with computer studies (e.g., computer scientist, software engineer, systems analyst), and the postsecondary education required to prepare for them	●							
D4.4 Evaluate their own development of Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport								