TFS High School

5635 Yong St. Suite 206,
Toronto, Ontario M2M 3S9

COURSE OUTLINE

Introduction to Computer Science University Preparation

ICS3U (University)

Department	Computer Studies
Instructor	Maliheh Mohseni
Course Development Date	September 2018
Ministry Course Code	ICS3U
Credit Value	1.00
Ministry Curriculum Document	Policy Document: Computer Studies, The Ontario Curriculum Grades 10 to 12, Revised 2008 http://www.edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf http://www.edu.gov.on.ca/eng/policyfunding/growSuccess.pdf
Prerequisites	None
Course Revision Date (TFS)	Dec 2020

COURSE DESCRIPTION

This course introduces students to computer science. Students will design software independently and as part of a team, using industry-standard programming tools and applying the software development life-cycle model. They will also write and use subprograms within computer programs. Students will develop creative solutions for various types of problems as their understanding of the computing environment grows. They will also explore environmental and ergonomic issues, emerging research in computer science, and global career trends in computer-related fields.

OVERALL EXPECTATIONS

Course Standards, Overall expectation	Concepts
A. Programming Concepts and Skills	A1. Data Types and Expressions A2. Control Structures and Simple Algorithms A3. Subprograms A4. Code Maintenance
B. Software Development	B1. Problem-solving Strategies B2. Designing Software Solutions B3. Designing Algorithms B4. The Software Development Life Cycle
C. Computer Environments and Systems	C1. Computer Components C2. File Maintenance C3. Software Development
D. Topics in Computer Science	D1. Environmental Stewardship and Sustainability D2. Exploring Computer Science D3. Postsecondary Opportunities

Specific Curriculum Expectations

A. Programming Concepts and Skills http://edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf
A1. Data Types and Expressions By the end of this course, students will: A1.1 use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs; A1.2 demonstrate an understanding of how a computer uses various systems *(e.g., binary, hexadecimal, ASCII, Unicode)* to internally represent data and store information; 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.
A2. Control Structures and Simple Algorithms By the end of this course, students will: 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).*

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

By the end of this course, students will:
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.

B SOFTWARE DEVELOPMENT

http://edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf

B1. Problem-solving Strategies

By the end of this course, students will:
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

By the end of this course, students will:
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, flowchart, UML [Unified Modeling Language], data flow diagram, pseudocode)*;
B2.5 design user-friendly software interfaces *(e.g., prompts, messages, screens, forms)*.

B3. Designing Algorithms

By the end of this course, students will:
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;
B3.3 design algorithms to detect, intercept, and handle exceptions *(e.g., division by zero, roots of negatives)*.

B4. The Software Development Cycle

By the end of this course, students will:
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;
B4.3 use project management tools *(e.g., Gantt chart, critical path diagram, PERT chart)* to show tasks and milestones in a teacher-led project;
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

http://edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf

C1. Computer Components

By the end of this course, students will:
C1.1 relate the specifications of the internal components of a computer *(e.g., CPU, RAM, ROM, cache, hard drive, motherboard, power supply, video card, sound card)* to user requirements;
C1.2 relate computer specifications *(e.g., processor type, bus speed, storage capacity, amount of memory)* to user requirements, using correct terminology;
C1.3 relate the specifications of common computer peripheral devices *(e.g., printer, monitor, scanner, keyboard, mouse, speakers, USB flash drive)* to user requirements;
C1.4 identify the computer components involved in executing programming operations *(e.g., assignment statements store a value in RAM, arithmetic operations are performed in the CPU).*

C2. File Maintenance

By the end of this course, students will:
C2.1 use an operating system to organize computer programs and files logically on local and shared drives;
C2.2 describe procedures to safeguard data and programs from malware *(e.g., viruses, Trojan horses, worms, spyware, adware, malevolent macros)*, and devise a thorough system protection plan;
C2.3 use standard procedures to back up and
archive user files.

C3. Software Development

By the end of this course, students will:
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;
C3.5 explain the difference between the functions of applications, programming languages, and operating systems.

D.TOPICS IN COMPUTER SCIENCE

http://edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf

D1. Environmental Stewardship and

By the end of this course, students will:
D1.1 describe the negative effects of computer use on the environment *(e.g., creation of e-waste, excessive use of paper resulting from unnecessary printing of files and emails, heavy, power consumption)* and on human health *(e.g., exposure to radiation, musculoskeletal disorders, eye strain, mental health problems resulting from social isolation, various health consequences of reduced activity levels)*;
D1.2 identify measures that help reduce the impact of computers on the environment *(e.g., lab regulations, school policies, corporate and government policies promoting paperless workplaces and computer recycling and reuse)* and on human health *(e.g., ergonomic standards)*;
D1.3 describe ways in which computers are or could be used to reduce resource use and to support environmental protection measures *(e.g., computer modelling to reduce use of physical resources; management of natural resources)*;
D1.4 identify government agencies and community partners that provide resources and guidance for environmental stewardship *(e.g., local community recycling centers, private companies that refurbish computers, printer cartridge recycling programs).*

D2. Exploring Computer Science

By the end of this course, students will:
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

By the end of this course, students will:
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.

Unit Titles and Descriptions

Course Contents:	Titles and Descriptions	Time
The Basic of Programming	**The Basics of Programing** The essential philosophies and logic surrounding programming, including models for input, output, processing, and all related terminology will be studies. Programming basics will be introduced. Simple programs will be constructed, using a number of different logical, calculation and algorithm strategies.	20 hours
Problem Solving with Procedures and Functions	**Introduction of Functions** Expanding upon the material covered in Unit I, students will develop more advanced programs, and delve into the real-life aspects of clarifying program specifications from clients, describing phases, milestones and products of software development, and the strategies behind debugging and troubleshooting.	22 hours
Computer Environments a	**The Computing Environment** In this unit, students will examine the fundamental aspects of the computing environment: hardware, specifications peripheral devices, software and applications, operating systems and basic programming codes and languages.	18 Hours
Software Development: **Using Data Structures** Advanced programming • Methods in Java (also called Subroutines, Subprograms, Functions, Procedures) • one-dimensional arrays • Strings	**Information Storage** Drawing upon knowledge from the course so far, students In this unit students learn how to create arrays, and how to write programs that declare, initialize, modify and access these arrays. Students will write algorithms with nested structures, and sub-programs, and algorithms that perform simple data management tasks. Use a variety of problem-solving strategies to solve different types of problems independently and as part of a team; Design software solutions to meet a variety of challenges; Design algorithms according to specifications; Apply a software development life-cycle model to a software development project.	35 hours
Project Students represent the stages of their project in the software development cycle. Its weight is %15 of Final grade.		5 hours
Final Evaluation The final assessment task will be comprised of two parts: a programming project representing the stages in the software development lifecycle and a final exam. Each of these two parts will		10 hours

constitute 15% of the final mark.	
Total	110 hours

TEACHING AND LEARNING STRATEGIES

The aim of this course is to introduce students to computer programming. In order to achieve this goal, a wide variety of instructional strategies are used to provide learning opportunities to accommodate a variety of learning styles, interests, and ability levels. The following are used throughout the course as strategies for teaching and learning the concepts presented:

- *Communicating:* Through the use of discussions, this course offers students the opportunity to share their understanding both in oral as well as written form.
- *Problem Solving:* This course scaffolds learning by providing students with the basic knowledge needed to understand computer science and building off of this knowledge as they progress through the course. The course guides students toward recognizing opportunities to apply knowledge they have gained to solve problems.
- *Connecting:* This course connects the concepts taught to real-world applications
- *Representing:* Through the use of examples, practice problems, and sample code, the course models various coding practices, poses questions that require students to use different representations as they are working at each level of conceptual development - concrete, visual or symbolic, and allows individual students the time they need to solidify their understanding at each conceptual stage.
- *Guided Exploration:* The course and teacher guide students through the exploration of a variety of coding practices and procedures necessary to be successful in computer science.

In addition, teacher and students have at their disposal a number of tools that are unique to electronic learning environments:
• Video presentations
• Discussion boards and google classroom
• Assessments with real-time feedback
• Interactive activities that engage both the student and teacher in the subject
• Peer review and assessment
• Internet Instructional Videos

The aim of this course is to prepare students with computer programming. Students are taught a lesson in zoom platform and then practice independently to apply what they have learned in different contexts as problem solving process.

ASSESSMENT/EVALUATION STRATEGIES:

Evaluations will consist of tests & quizzes, assignments, projects, group work, and presentations.

To promote student success, ongoing formative assessment and feedback will be given to students. The course expectations will be evaluated according to the four categories of the achievement chart.

The evaluation for this course is based on the student's achievement of curriculum expectations and the demonstrated skills required for effective learning. The final percentage grade represents the quality of the student's overall achievement of the expectations for the course and reflects the corresponding level of achievement as described in the achievement chart for the discipline. A credit is granted and recorded for this course if the student's grade is 50% or higher.

Diagnostic assessment is used at the beginning of a unit to assist in determining a starting point for instruction. Assessment for Learning (AFL) provides information to students as they are learning and refining their skills. Assessment as Learning (AAL) acts as a stepping-stone for students to begin applying their understanding using critical thinking; it bridges the gap between AFL and AOL. Assessment of Learning (AOL), at the end of units and course, provides students with the opportunity to synthesize/apply/demonstrate their learning and the achievement of the expectations.
Evaluation is based on gathering evidence of student achievement through:

- Products
- Observations
- Conversations

The following is a list of specific assessment/evaluation strategies that the teacher may use but is not limited to:

Evaluation & Additional Information Evaluation in this course will be diagnostic, formative and summative.			
Summative Evaluation	70%	Knowledge and Understanding	25%
		Thinking/Inquiry & Problem-solving	25%
		Application	25%
		Communication	25%
Final Evaluation	30%	project	15%
		Examination	15%

Rating Scale	
Excellent	80 - 100 %
Good	66 - 79 %
Satisfactory	50 - 65 %
Needs Improvement	Below 50%

Teaching and Learning Strategies:

Communication	• Report/Presentation	• Collaborative/Cooperative

Problem Solving	• **Homework**	• Software Life Cycle Design Process
Connecting to real-world	• **Critical thinking**	• Computer-based Tutorials/Exploration Activities
Guided exploration	• **Brainstorming**	• Independent Study

Learning Skills:

Learning Skills are skills and habits are essential to success in school and in the workplace. Teachers report achievement on the six Learning Skills shown below the table below using letter codes:

♦Responsibility ♦Organization ♦Independent Work ♦Collaboration ♦Initiative ♦ Self-regulation

E = Excellent G = Good S = Satisfactory N = Needs Improvement.

Learning Skills	Sample Behaviors
Responsibility	The student fulfils responsibilities and commitments within the learning environment; completes and submits class work, homework, and assignments according to agreed-upon timelines; takes responsibility for and manages own behavior.
Organization	The student devises and follows a plan and process for completing work and tasks; establishes priorities and manages time to complete tasks and achieve goals; identifies, gathers, evaluates, and uses information, technology, and resources to complete tasks.
Independent Work	The student independently monitors, assesses, and revises plans to complete tasks and meet goals; uses class time appropriately to complete tasks; follows instructions with minimal supervision.
Collaboration	The student accepts various roles and an equitable share of work in a group; responds positively to the ideas, opinions, values, and traditions of others; builds healthy peer-to-peer relationships through personal and media-assisted interactions; works with others to resolve conflicts and build consensus to achieve group goals; shares information, resources, and expertise and promotes critical thinking to solve problems and make decisions.
Initiative	The student looks for and acts on new ideas and opportunities for learning; demonstrates the capacity for innovation and a willingness to take risks; demonstrates curiosity and interest in learning; approaches new tasks with a positive attitude; recognizes and advocates appropriately for the rights of self and others.
Self-Regulation	The student sets own individual goals and monitors progress towards achieving them; seeks clarification or assistance when needed; assesses and reflects critically on own strengths, needs, and interests; identifies learning opportunities, choices, and strategies to meet personal needs and achieve goals; perseveres and makes an effort when responding to challenges.

STRATEGIES FOR ASSESSMENT AND EVALUATION OF STUDENT PERFORMANCE

The tools below will be used for the three different types of assessments:

Assessment as Learning

Student Product

- Learning logs

Observation

- Whole class discussions

Conversation

- Student teacher discussions

- Small group discussions

- Pair works

Assessment for Learning

Student Product

- Assignment

- Pre-tests (scale/rubric)

- Quizzes (scale/rubric)

Observation

- Class discussions

- PowerPoint presentations (rubric)

Conversation

- Student teacher conferences (checklist)

- Small group discussions (checklist)

- Pair work (checklist)

Assessment of Learning

Student Product

- Assignment

- Tests (scale/rubric)

- Exam (scale/rubric)

- Project (rubric/scale)

Observation

- PowerPoint presentations (rubric)

- Performance tasks (rubric/scale)

Conversation

- Student teacher conferences (checklist)

- Question and answer session (checklist)

- Oral tests (scale/rubric)

Assessment Methods and Tools:

Term Assessment and Evaluation: 70% (Tests, Exams, Assignments, Projects)

Knowledge and Understanding (25%): Knowledge of content (e.g., facts, terms, definitions and procedures.) Understanding of content (e.g., concepts, principles, theories, relationships and methodologies)

Thinking and Inquiry (25%): Planning skills (e.g., focusing research, gathering information, selecting strategies, organizing a project) Processing skills (e.g., analyzing, interpreting, assessing, reasoning, gathering ideas, evaluating, seeking a variety of perspectives, forming conclusions)

Communication (25%): Expression of original ideas and information (e.g., logical organization) in oral, visual, and written forms

Application/Making connection (25%): The use of the knowledge and skills to make connections within and between various contexts.

Categories	50-59% (Level 1)	60-69% (Level 2)	70-79% (Level 3)	80-100% (Level 4)
Knowledge and Understanding - Subject-specific content acquired in each course (knowledge), and the comprehension of its meaning and significance (understanding)				
	The student:			
Knowledge of content (e.g., facts, terms, procedural skills, use of tools)	demonstrates limited knowledge of content	demonstrates some knowledge of content	demonstrates considerable knowledge of content	demonstrates thorough knowledge of content
Understanding of mathematical concepts	demonstrates limited understanding of content	demonstrates some understanding of content	demonstrates considerable understanding of content	demonstrates thorough and insightful understanding of content
Thinking - The use of critical and creative thinking skills and/or processes				
	The student:			
Use of planning skills -understanding the problem (e.g., formulating and interpreting the problem, making conjectures)	uses planning skills with limited effectiveness	uses planning skills with moderate effectiveness	uses planning skills with considerable effectiveness	uses planning skills with a high degree of effectiveness

-making a plan for problem solving				
Use of processing skills -carrying out a plan (e.g., collecting data, questioning, testing, revising, modelling, solving, inferring, forming conclusions) -looking back at the solution (e.g., evaluating reasonableness, making convincing arguments, reasoning, justifying, proving, reflecting)	uses processing skills with limited effectiveness	uses processing skills with some effectiveness	uses processing skills with considerable effectiveness	uses processing skills with a high degree of effectiveness
Use of critical/creative thinking processes (e.g., problem solving, inquiry)	uses critical / creative thinking processes with limited effectiveness	uses critical / creative thinking processes with some effectiveness	uses critical / creative thinking processes with considerable effectiveness	uses critical / creative thinking processes with a high degree of effectiveness
Communication - The conveying of meaning through various forms				
	The student:			
Expression and organization of ideas and mathematical thinking (e.g., clarity of expression, logical organization), **using oral, visual, and written forms** (e.g., pictorial, graphic, dynamic, numeric, algebraic forms; concrete materials)	expresses and organizes mathematical thinking with limited effectiveness	expresses and organizes mathematical thinking with some effectiveness	expresses and organizes mathematical thinking with considerable effectiveness	expresses and organizes mathematical thinking with a high degree of effectiveness
Communication for different audiences (e.g., peers and teachers) **and purposes** (e.g., to present data, justify a solution, express a mathematical argument) **in oral, visual, and written forms**	communicates for different audiences and purposes with limited effectiveness	communicates for different audiences and purposes with some effectiveness	communicates for different audiences and purposes with considerable effectiveness	communicates for different audiences and purposes with a high degree of effectiveness
Use of conventions, vocabulary, and terminology of the discipline (e.g., terms, symbols) **in oral, visual, and written forms**	uses conventions, vocabulary, and terminology of the discipline with limited effectiveness	uses conventions, vocabulary, and terminology of the discipline with some effectiveness	uses conventions, vocabulary, and terminology of the discipline with considerable effectiveness	uses conventions, vocabulary, and terminology of the discipline with a high degree of effectiveness
Application - The use of knowledge and skills to make connections within and between various contexts				

	The student:			
Application of knowledge and skills in familiar contexts	applies knowledge and skills in familiar contexts with limited effectiveness	applies knowledge and skills in familiar contexts with some effectiveness	applies knowledge and skills in familiar contexts with considerable effectiveness	applies knowledge and skills in familiar contexts with a high degree of effectiveness
Transfer of knowledge and skills to new contexts	transfers knowledge and skills to new contexts with limited effectiveness	transfers knowledge and skills to new contexts with some effectiveness	transfers knowledge and skills to new contexts with considerable effectiveness	transfers knowledge and skills to new contexts with a high degree of effectiveness
Making connections within and between various contexts (e.g., connections between concepts, representations, and forms within mathematics; connections involving use of prior knowledge and experience; connections between mathematics, other disciplines, and the real world))	makes connections within and between various contexts with limited effectiveness	makes connections within and between various contexts with some effectiveness	makes connections within and between various contexts with considerable effectiveness	makes connections within and between various contexts with a high degree of effectiveness

Resources required by the student:

Java Development Kit (JDK) from Oracle (A link to download this software for free is provided in the course).

NetBeans IDE (A link to download this free software for Mac or Windows is provided in the course)

RESOURCES

Ontario Ministry of Education (EDU) – curriculum documents page

http://www.edu.gov.on.ca/eng/document/curricul/curricul.html

http://www.edu.gov.on.ca/eng/policyfunding/growSuccess.pdf

Study Guides

Note: This course is entirely remote learning.