Introduction to Computer Science, Grade 11

University Preparation — ICS3U

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.

Prerequisite: None

A. PROGRAMMING CONCEPTS AND SKILLS

OVERALL EXPECTATIONS

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

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

By the end of this course, students will:

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

OVERALL EXPECTATIONS

By the end of this course, students will:

B1. use a variety of problem-solving strategies to solve different types of problems independently and as part of a 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.

SPECIFIC EXPECTATIONS

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, 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

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 Life 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

OVERALL EXPECTATIONS

By the end of this course, students will:

C1. relate the specifications of computer components to user requirements;

C2. use appropriate file maintenance practices to organize and safeguard data;

C3. demonstrate an understanding of the software development process.

SPECIFIC EXPECTATIONS

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

OVERALL EXPECTATIONS

By the end of this course, students will:

D1. describe policies on computer use that promote environmental stewardship and sustainability;

D2. demonstrate an understanding of emerging areas of computer science research;

D3. describe postsecondary education and career prospects related to computer studies.

SPECIFIC EXPECTATIONS

D1. Environmental Stewardship and Sustainability

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 centres, 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.