Computer Science, Grade 12

University Preparation — ICS4U

This course enables students to further develop knowledge and skills in computer science. Students will use modular design principles to create complex and fully documented programs, according to industry standards. Student teams will manage a large software development project, from planning through to project review. Students will also analyse algorithms for effectiveness. They will investigate ethical issues in computing and further explore environmental issues, emerging technologies, areas of research in computer science, and careers in the field.

Prerequisite: Introduction to Computer Science, Grade 11, University Preparation

A. PROGRAMMING CONCEPTS AND SKILLS

OVERALL EXPECTATIONS

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

A1. Data Types and Expressions

By the end of this course, students will:

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.

A2. Modular Programming

By the end of this course, students will:

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

By the end of this course, students will:

A3.1 demonstrate the ability to read from, and write to, an external file *(e.g., text file, binary file, database, 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)*.

A4. Code Maintenance

By the end of this course, students will:

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

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

B1. Project Management

By the end of this course, students will:

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;

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

By the end of this course, students will:

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

By the end of this course, students will:

C1. demonstrate the ability to apply modular design concepts in computer programs;

C2. analyse algorithms for their effectiveness in solving a problem.

SPECIFIC EXPECTATIONS

C1. Modular Design

By the end of this course, students will:

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

By the end of this course, students will:

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

By the end of this course, students will:

D1. assess strategies and initiatives that promote environmental stewardship with respect to the use of computers and related technologies;

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.

SPECIFIC EXPECTATIONS

D1. Environmental Stewardship and Sustainability

By the end of this course, students will:

D1.1 outline strategies to reduce the impact of computers and related technologies on the environment *(e.g., reduce, reuse, and recycle; turn computers and monitors off at end of day; participate in printer cartridge recycling)* and on human health *(e.g. ergonomic standards)*;

D1.2 investigate and report on governmental and community initiatives that encourage environmental stewardship and promote programs and practices that support sustainability *(e.g., local community recycling centres, private companies that refurbish computers, printer cartridge recycling programs)*.

D2. Ethical Practices

By the end of this course, students will:

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)*;

D2.3 outline and apply strategies to encourage ethical computing practices at home, at school, and at work.

D3. Emerging Technologies and Society

By the end of this course, students will:

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

By the end of this course, students will:

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.