Introduction to Computer Science, Grade 11, University Preparation (ICS3U)

TDSB – Forest Hill Collegiate Institute (416 393–1860)

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 sub-programs 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.
(Computer Studies: The Ontario Curriculum Grades 10 to 12, Revised, 2008) **Prerequisite**: None

CURRICULUM & OVERALL EXPECTATIONS

Course Strands	*Concepts*
A. Programming Concepts & Skills	A1. Data Types, Variables, Arithmetic Operations & Expressions A2. Control Structures and Simple Algorithms A3. Modularity (Sub-programs / Sub-routines) A4. Code Maintenance
B. Software Development	B1. Problem-solving Strategies B2. Designing Software Solutions (including graphics) B3. Designing Algorithms B4. The Software Development Life Cycle
C. Computer Environments & 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

ASSESSMENT & EVALUATION

Evaluations will consist of traditional tests & quizzes, assignments, projects, group work, and presentations.
Students can expect at least one written evaluation (quiz, test) and at least one major assignment or project per unit.
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.

Term Work / Evaluations		
Knowledge & Understanding	20%	
Application	30%	
Thinking/Inquiry & Problem-solving (TIPS)	30%	***70 to 90%**
Communication	20%	
Course Culminating Evaluations		
Final Programming Project	15%	***10 to 30%**
Final Written Test	15%	
Final Course Mark		**100%**

****tentative; subject to change based on TDSB and Ministry guidelines and policy adjustments***

PROVISIONS FOR STUDENT SUCCESS

Teacher Support:	**Student Responsibilities:**
• *Extra help* • *Computer Lab hours* *During lunch or after school* *(arrange time with teacher in advance)*	• Keep an organized binder to assist with your studies • Keep an organized "digital binder" of all digital materials provided • Set and focus on realistic goals for each class, each unit, the whole course • Record daily achievements to set and meet new challenges • Review learning at home daily • Home study in preparation for each class to enrich your learning • Take advantage of extra help and lab hours to assist in meeting goals • Provide help to peers to consolidate your learning and increase confidence • Ask and answer questions; look hard for answers; participate!

LEARNING SKILLS ASSESSMENT

These skills will be assessed throughout the course and reflected on the report card.

Learning Skills and Work Habits	E – Excellent G – Good S – Satisfactory N – Needs Improvement			
Responsibility			**Organization**	
▪ 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 behaviour.			▪ 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			**Collaboration**	
▪ Independently monitors, assesses, and revises plans to complete tasks and meet goals. ▪ Uses class time appropriately to complete tasks. ▪ Follows instructions with minimal supervision.			▪ 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			**Self-Regulation**	
▪ 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.			▪ 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.	

For further details, see "*Growing Success*" (p. 9 – 14): http://www.edu.gov.on.ca/eng/policyfunding/growSuccess.pdf

ONTARIO SKILLS PASSPORT (OSP)

- Essential skills needed for work, learning, and life: Reading text, Writing, Computer Use, Measurement and Calculation, and Problem Solving (http://www.skills.edu.gov.on.ca/OSP2Web/EDU/Welcome.xhtml)

POLICIES & CLASSROOM EXPECTATIONS

- ***Academic Honesty*** – Students are expected to be academically honest. They are expected to submit their own work, so that the mark received reflects their own academic achievement.
- ***Lab Rules*** – computers and equipment are to be used for course-related work (see Acceptable Use Policy for more details)
- ***Care & Respect for Property*** – no food or drink in the lab, take care of equipment, report any issues/problems to teacher
- ***Online Code of Conduct*** as in the school agenda or http://www.tdsb.on.ca/communications/code_of_online_conduct/occ.html
- ***Assignments & Projects*** – assignments and projects are to be completed and submitted by the due date and are subject to late penalties (work submitted after the ultimate deadline will not be accepted and will receive a mark of zero)

RESOURCES, COURSE MATERIALS, USEFUL WEBSITES (PROGRAMMING & ONLINE TUTORIALS)

- USB flash drive or external HDD for storing and transporting digital material and work between school and home
- 3-ring binder, paper, writing utensils (pen, pencil, eraser)
- *Ready to Program Java IDE (http://compsci.ca/holtsoft/)*
- *Dr. Java IDE (http://drjava.org/)*
- *Eclipse IDE (https://www.eclipse.org/)*
- *Google classroom*
- *Textbooks: various (several texts are available as resource/reference in classroom and in digital form)*
- *Chortle Online Java Tutorial (http://chortle.ccsu.edu/CS151/cs151java.html)*
- *Programming Using Java (an online text) (http://math.hws.edu/javanotes/)*
- *Oracle Java Tutorial (http://download.oracle.com/javase/tutorial/)*
- *The Java API (http://download.oracle.com/javase/1.4.2/docs/api/)*

Introduction to Computer Science (ICS3U) – Course Content by Unit

Unit 1: Intro to Computers

Some Basics

- Lab Rules, AUP
- File & Folder Management – backing up and organizing computer files
- Keeping a Glossary, Programming Reference Guide

Hardware & Software, Networks, etc

- Four functions of a computer
- Input, Output, and Storage Devices
- Computer Peripherals (keyboard, mouse, printer, monitor, scanner, speakers, etc)
- System & Application Software
- Operating Systems
- Boot-up Process (BIOS, POST, OS)
- Internal components of a computer (CPU, RAM, ROM, motherboard, etc)
- Computer Specifications (processor, bus speed, storage capacity, etc)
- Fetch-Decode-Execute Cycle (CPU and RAM)
- Network Size, Architecture, Topology
- Emerging Technologies & Computer Issues

Unit 2: Intro to Programming

The Basics of Programming Languages

- Integrated Development Environments (IDEs), Getting "Ready to Program" (installing Java IDEs and necessary files at home)
- System & Application Software, Operating Systems, and Programming Languages
- Binary/Hexadecimal/ASCII/Unicode
- Source code vs. Machine code
- Interpreters vs. Compilers
- High-level vs. Low-level Languages
- Structured vs. Object-oriented Programming

The IPO Process/Model – Basic Output, Arithmetic Operators & Expressions, Data Types, Variables & Constants, Input, etc.

- Basic Output (character/text, graphics)
- Data Types, Variables, and Constants (integers, floating point, boolean, characters, Strings, etc.)
- Assigning values to constants/variables
- Input from the keyboard
- Identify and correct basic syntax, logic and run-time errors, and interpret basic error messages (compile-time vs run-time errors)
- Interpret basic error messages
- Arithmetic Operators & Expressions
- Formatting Output, Rounding
- Integer division, truncation
- Casting
- Concatenation

Unit 3: Control Structures

- Selection structures (if statements, switch statements, etc.)
- Relational (comparison) operators
- Repetition structures (conditional loops, counted loops, etc.)
- Boolean (logic) operators: AND, OR, NOT, XOR
- Tracing program flow to correct errors
- IPO Model
- Nested structures, nested loops
- Solve common problems involving mathematics
- User-friendly interfaces

Unit 4: Software Development Life Cycle

This unit covers the software development life cycle (SDLC) and also issues with malware. This unit is covered during the whole semester; it is woven/delivered throughout the other units.

The Software Development Life Cycle (Process):

1. **Analysis (Requirements/Specifications) Stage**
   - clarifying program specifications using various tools and techniques (dialogue, questionnaires, surveys, research, program description, IPO chart, list of knowns & unknowns)
2. **Design & Development Stage**
   - represent structure and components of a program (pseudo-code, flowchart, structure chart, UML [Unified Modeling Language], data flow diagram)
3. **Implementation Stage**
   - Use proper programming conventions (variable names, indenting the code, commenting the code)
4. **Testing & Debugging Stage**
   - Use a test plan with test cases to test programs (identify test scenarios, identify suitable input data, calculate expected outcomes, record actual outcomes, and conclude "pass" or "fail") recording
   - Use a variety of methods to debug programs (output statements added to code, debugger, trace program flow)
5. **Maintenance Stage**
6. **Documentation Stage**

All Stages:

- status reporting in terms of work completed, work outstanding, and milestones reached
- proper exception handling should be considered and coded to detect, intercept, and handle exceptions (division by zero, roots of negatives)
- show tasks and milestones in project management tools (Gantt chart, critical path diagram, PERT chart) in a teacher-led project

Malware:

- describe procedures to safeguard data and programs from all types of malware (viruses, Trojan horses, worms, spyware, adware, malevolent macros, spambots, denial of service (DDoS), …)

Unit 5: Modularity (Methods)

Structuring programs and designing re-usable code. Students will learn to write modular programs by designing and implementing their own methods (functions), and improve their ability to solve problems both independently and as part of a small team. Continued use of appropriate conventions and standards, along with appropriate terminology and vocabulary.

Topics & Concepts:

1. complete a program using a program template or skeleton

2. use built-in / existing methods (functions):
- random number generator
- string methods (e.g. substring)
- math (e.g. sqrt, absolute value, trig.)
- powers / exponents

3. write methods (functions) incorporating:
- passing parameters (parameter vs. argument)
- returning values
- local versus global scope of variables

4. explain the structure of a program and describe alternative program designs

Unit 6: Indexed Data Structures (Arrays)

Students can work individually or in a team setting to cover these concepts. Use arrays to teach simple algorithms and more complex nested structures. Various problem-solving strategies can be taught *(e.g., stepwise refinement, divide and conquer, working backwards, examples, extreme cases, tables and charts, trial and error)*.

All about 1D arrays:
- elements, indexes, and bounds
- declare and initialize arrays
- access array elements
- modifying array elements

Some concepts:
- count elements in an array
- calculate total, highest, lowest value
- linear search
- add sorted array
- deleting datum from the middle of an array

Unit 7: Computers in Society & the Environment

Effects of computer use on the environment
- e-waste/excessive use of paper/power consumption
- measures that help reduce impact of computers on environment (policies, paperless workplaces etc)
- ways in which computers could be used to reduced resource use (computer modelling to reduce use of physical resources)
- government agencies/community partners that provide resources/guidance for environmental stewardship

Effects of computer use on human health
- Radiation/musculoskeletal disorders/ergonomic issues etc.
- mental health issues/social isolation/issues relating to reduced activity levels

Exploring Computer Science Topics & Issues
- emerging areas of research in computer science: cryptography, parallel processing, human-computer interaction, artificial intelligence, GIS, security, etc.
- understanding collaborative research between computer science and another field (e.g. bioinformatics)

Post-secondary Opportunities
- career choices and trends in computer science (local/national/international levels)
- identify experiential learning opportunities in computer science (co-op, job shadowing, etc.)
- post-secondary educational opportunities leading to careers in information science; university/college/industry certifications, etc.