2008

REVISED

The Ontario Curriculum
Grades 10 to 12

Computer Studies

reach every student

Ontario

CONTENTS

Une publication équivalente est disponible en français sous le titre suivant :
Le curriculum de l'Ontario, de la 10e à la 12e année : Études informatiques.

This publication is available on the Ministry of Education's website, at www.edu.gov.on.ca.

COURSES 31

INTRODUCTION

This document replaces the Computer and Information Science component of *The Ontario Curriculum, Grades 9 and 10: Technological Education, 1999,* and of *The Ontario Curriculum, Grades 11 and 12: Technological Education, 2000*. Beginning in September 2009, all computer studies courses for Grades 10 to 12 will be based on the expectations outlined in this document.

SECONDARY SCHOOLS FOR THE TWENTY-FIRST CENTURY

The goal of Ontario secondary schools is to support high-quality learning while giving individual students the opportunity to choose programs that suit their skills and interests. The updated Ontario curriculum, in combination with a broader range of learning options outside traditional classroom instruction, will enable students to better customize their high school education and improve their prospects for success in school and in life.

THE IMPORTANCE OF COMPUTER STUDIES IN THE CURRICULUM

Computer studies is about how computers compute. It is not about learning how to use the computer, and it is much more than computer programming. Computer studies is the study of ways of representing objects and processes. It involves defining problems; analysing problems; designing solutions; and developing, testing, and maintaining programs. For the purposes of this document, the term *computer studies* refers to the study of computer science, meaning computer and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society. The major focus of these courses is the development of programming skills, which are important for success in future postsecondary studies.

Computer studies is relevant for all students because it incorporates a broad range of transferable problem-solving skills and techniques, including logical thinking, creative design, synthesis, and evaluation. It also teaches generically useful skills in such areas as communication, time management, organization, and teamwork. Students live in a technologically rich world, and computer studies will provide them with the knowledge and skills to understand the underpinnings of current computer technology and prepare them for emerging technologies. A foundation in this discipline will introduce students to the excitement and opportunities afforded by this dynamic field and will begin to prepare them for a range of rewarding careers.

The computer studies program will build a strong foundation for those who wish to move on to further study and training in specialized areas such as computer programming, database analysis, computer science, education, computer engineering, software engineering, information technology, and game development.

THE GOALS OF THE COMPUTER STUDIES PROGRAM

The fundamental purpose of the computer studies program is to provide students with knowledge, skills, and attitudes that will enable them to achieve success in secondary school, the workplace, postsecondary education or training, and daily life.

The goals of the computer studies curriculum are to enable students to:

- gain an understanding of computer studies concepts;
- develop the skills, including critical thinking skills, and the knowledge of strategies required to do research, conduct inquiries, and communicate findings accurately, ethically, and effectively;
- apply the knowledge, skills, and attitudes acquired through the study of computers to a variety of learning tasks and relate them to computer phenomena on the local, national, and global levels;
- develop lifelong learning habits that will help them adapt to computer advances in the changing workplace and world;
- make connections that will help them take advantage of potential postsecondary educational and work opportunities.

FOUR CRITICAL AREAS OF LEARNING IN COMPUTER STUDIES

Effective learning in all aspects of computer studies depends on the development of knowledge and skills in the following four areas:

- Software development (including project management and software engineering principles)
- Algorithms and data structures
- Program correctness and efficiency
- Professional and ethical responsibility

The computer studies curriculum in Grades 10 to 12 offers a range of courses, all of which are structured to give students a solid foundation in these four areas.

ROLES AND RESPONSIBILITIES IN COMPUTER STUDIES PROGRAMS

Students

Students have many responsibilities with regard to their learning. Students who make the effort required to succeed in school and who are able to apply themselves will soon discover that there is a direct relationship between this effort and their achievement, and will therefore be more motivated to work. There will be some students, however, who will find it more difficult to take responsibility for their learning because of special challenges they face. The attention, patience, and encouragement of teachers can be extremely important to the success of these students. However, taking responsibility for their own progress and learning is an important part of education for all students, regardless of their circumstances.

Mastering the concepts and skills connected with computer studies requires work, study, and the development of cooperative skills. In addition, students who actively pursue opportunities outside the classroom will extend and enrich their understanding of computers and related topics and issues. Their understanding and skills will grow as they engage in recreational activities that involve computer use (e.g., computer clubs), reading related to computers (e.g., magazines, Internet sources), and learning about advances in computer studies (e.g., attending computer fairs).

Parents

Parents[1] have an important role to play in supporting student learning. Studies show that students perform better in school if their parents are involved in their education. By becoming familiar with the curriculum, parents can determine what is being taught in the courses their daughters and sons are taking and what they are expected to learn. This awareness will enhance parents' ability to discuss their children's work with them, to communicate with teachers, and to ask relevant questions about their children's progress. Knowledge of the expectations in the various courses will also help parents to interpret teachers' comments on student progress and to work with teachers to improve their children's learning.

Effective ways in which parents can support their children's learning include attending parent-teacher interviews, participating in parent workshops, becoming involved in school council activities (including becoming a school council member), and encouraging students to complete their assignments at home. In addition to supporting regular school activities, parents may wish to provide their daughters and sons with opportunities to question and reflect on current affairs, including developments in the field of emerging technologies.

Teachers

Teachers and students have complementary responsibilities. Teachers develop appropriate instructional strategies to help students achieve the curriculum expectations, as well as appropriate methods for assessing and evaluating student learning. Teachers also support students in developing the reading, writing, oral communication, and numeracy skills needed for success in all their courses. Teachers bring enthusiasm and varied teaching and assessment approaches to the classroom, addressing different student needs and ensuring sound learning opportunities for every student.

Using a variety of instructional, assessment, and evaluation strategies, teachers provide numerous opportunities for students to develop a range of skills and knowledge, including knowledge of computer studies concepts, structures, and processes, that will allow them to participate more effectively in their communities as responsible and active citizens.

Principals

The principal works in partnership with teachers and parents to ensure that each student has access to the best possible educational experience. To support student learning, principals ensure that the Ontario curriculum is being properly implemented in all classrooms using a variety of instructional approaches. They also ensure that appropriate resources are made available for teachers and students. To enhance teaching and learning in all subjects, including computer studies, principals promote learning teams and

1. The word *parents* is used in this document to refer to parent(s) and guardian(s).

work with teachers to facilitate their participation in professional development activities. Principals are also responsible for ensuring that every student who has an Individual Education Plan (IEP) is receiving the modifications and/or accommodations described in his or her plan – in other words, for ensuring that the IEP is properly developed, implemented, and monitored.

Community Partnerships

Community partners in the area of computer studies can be an important resource for schools and students. They can provide support for students in the classroom, and can be models of how the knowledge and skills acquired through study of the curriculum relate to life beyond school. As mentors, they can enrich not only the educational experience of students, but also the life of the community. Schools can, for example, make arrangements with firms in the community to provide computer specialists for in-class workshops for students based on topics, concepts, and skills from the curriculum, as well as to create opportunities for cooperative education.

Schools and school boards can play a role by coordinating efforts with community partners. They can involve colleges, universities, trade unions or professional organizations, local businesses, and community volunteers in supporting instruction and in promoting a focus on computer studies in and outside the school. Postsecondary institutions and other community stakeholders can be included in events held at the school (such as parent education nights, programming skills competitions, and joint ventures), and school boards can collaborate with their community partners by providing educational opportunities within the community.

THE PROGRAM IN COMPUTER STUDIES

OVERVIEW OF THE PROGRAM

The computer studies program comprises courses in Grades 10, 11, and 12. Three types of courses are offered in the program: university preparation, college preparation, and open courses. Students choose between course types on the basis of their interests, achievement, and postsecondary goals. The course types are defined as follows:

- ***University preparation*** *courses are designed to equip students with the knowledge and skills they need to meet the entrance requirements for university programs.*
- ***College preparation*** *courses are designed to equip students with the knowledge and skills they need to meet the requirements for entrance to most college programs or for admission to apprenticeship or other training programs.*
- ***Open*** *courses are designed to broaden students' knowledge and skills in subjects that reflect their interests and to prepare them for active and rewarding participation in society. They are not designed with the specific requirements of universities, colleges, or the workplace in mind.*

The program in computer studies offers a variety of courses to help students develop a deeper understanding of the world of computer science and computer programming, and to help focus their interests in this area.

The Grade 10 open course will appeal to any student interested in exploring how computers are used to solve problems. The course provides students with the opportunity to develop the logical thinking processes used in designing computer solutions to problems and to acquire basic computer programming skills that will enable them to create a working computer program.

College preparation courses focus on the development of computer programming skills. These courses introduce students to the types of programs offered at community colleges – programs that focus on the practical skills businesses currently demand. Students are given opportunities to use problem-solving strategies and tools to address challenges such as creating custom programs, tailoring existing program packages, and using database management systems and scripting languages.

University preparation courses provide students with a foundation in the broad spectrum of computer science concepts and applications. In these courses, students explore the theoretical underpinnings of computer science; design software, working independently

and as part of a team and using industry-standard programming tools and the software development life-cycle model; and investigate various computer science–related topics, such as cryptography and artificial intelligence.

The computer studies program is designed to enable students to select courses that relate to their interests and that will prepare them for further study or work in the field of their choosing. Computer studies courses are well suited for inclusion in programs that lead to a diploma with a Specialist High Skills Major designation. Whether students eventually work in the computer field or simply use computers, the computer studies program will provide them with a foundation for making informed decisions about their future plans.

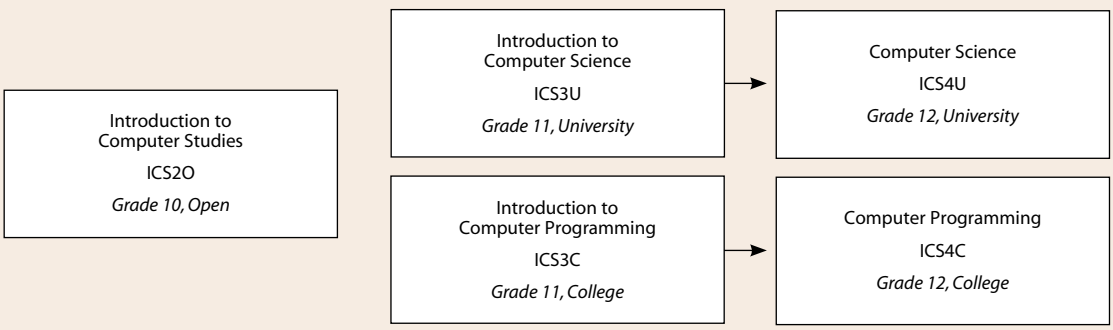
Courses in Computer Studies, Grades 10–12

Grade	Course Name	Course Type	Course Code	Prerequisite
10	Introduction to Computer Studies	Open	ICS2O	None
11	Introduction to Computer Science	University	ICS3U	None
11	Introduction to Computer Programming	College	ICS3C	None
12	Computer Science	University	ICS4U	Grade 11 Introduction to Computer Science, University
12	Computer Programming	College	ICS4C	Grade 11 Introduction to Computer Programming, College

Note: Each of the courses listed above is worth one credit.

Prerequisite Chart for Computer Studies, Grades 10–12

This chart maps out all the courses in the discipline and shows the links between courses and the possible prerequisites for them.



Although courses in computer studies are optional, students should keep in mind that they can take any computer studies course in the Grade 10–12 program to fulfil the Group 3 additional compulsory credit requirement for the Ontario Secondary School Diploma.[2]

Half-Credit Courses

The courses outlined in this curriculum document are designed as full-credit courses. However, *with the exception of the Grade 12 university preparation course*, they may also be delivered as half-credit courses.

Half-credit courses, which require a minimum of fifty-five hours of scheduled instructional time, must adhere to the following conditions:

- The two half-credit courses created from a full course must together contain all of the expectations of the full course. The expectations for each half-credit course must be drawn from all strands of the full course and must be divided in a manner that best enables students to achieve the required knowledge and skills in the allotted time.
- A course that is a prerequisite for another course in the secondary curriculum may be offered as two half-credit courses, but students must successfully complete both parts of the course to fulfil the prerequisite. (Students are not required to complete both parts unless the course is a prerequisite for another course they wish to take.)
- The title of each half-credit course must include the designation *Part 1* or *Part 2*. A half credit (0.5) will be recorded in the credit-value column of both the report card and the Ontario Student Transcript.

Boards will ensure that all half-credit courses comply with the conditions described above, and will report all half-credit courses to the ministry annually in the School October Report.

CURRICULUM EXPECTATIONS

The expectations identified for each course describe the knowledge and skills that students are expected to develop and demonstrate in their class work, on tests, and in various other activities on which their achievement is assessed and evaluated.

Two sets of expectations are listed for each *strand*, or broad curriculum area, of each course. (The strands are numbered A, B, C, and so on.)

- The *overall expectations* describe in general terms the knowledge and skills that students are expected to demonstrate by the end of each course.
- The *specific expectations* describe the expected knowledge and skills in greater detail. The specific expectations are grouped under numbered subheadings, each of which indicates the strand and the overall expectation to which the subgrouping of specific expectations corresponds (e.g., "B2" indicates that the group relates to overall expectation 2 in strand B). The subheadings may serve as a guide for teachers as they plan learning activities for their students.

2. To meet the Group 3 additional compulsory credit requirement, students have the choice of earning one credit for a course in computer studies (Grades 10 to 12) or technological education (Grades 9 to 12), or one credit for an additional course in science (Grade 11 or 12), or one credit for a cooperative education course.

Each computer studies course is organized into three or four **strands**, numbered A, B, C, and so on.

The **overall expectations** describe in general terms the knowledge and skills students are expected to demonstrate by the end of each course. Three or more overall expectations are provided for each strand in every course. The numbering of overall expectations indicates the strand to which they belong (e.g., B1 through B4 are the overall expectations for strand B).

Grade 11, University Preparation

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.

A **numbered subheading** identifies each group of specific expectations and relates to one particular overall expectation (e.g., "B2. Designing Software Solutions" relates to overall expectation B2).

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;

The **specific expectations** describe the expected knowledge and skills in greater detail. The expectation number identifies the strand to which the expectation belongs and the overall expectation to which it relates (e.g., B2.1, B2.2, B2.3, and so on, relate to the second overall expectation in strand B).

Examples are introduced by "e.g." and appear in parentheses, set in italics, within specific expectations. The examples help to clarify the requirement specified in the expectation and to suggest its intended depth and level of complexity. The examples are illustrations only, not requirements.

The organization of expectations into strands and subgroupings is not meant to imply that the expectations in any one strand or group are achieved independently of the expectations in the other strands or groups. The strands and subgroupings are used merely to help teachers focus on particular aspects of knowledge and skills as they develop and present various lessons and learning activities for their students. The concepts, content, and skills identified in the different strands of each course should, wherever appropriate, be integrated in instruction throughout the course.

Many of the specific expectations are accompanied by examples, which are given in parentheses and italicized. These examples are meant to illustrate the kind of knowledge or skill, the specific area of learning, the depth of learning, and/or the level of complexity that the expectation entails. The examples are intended as a guide for teachers rather than as an exhaustive or mandatory list.

Teachers can choose to use the examples that are appropriate for their classrooms or they may develop their own approaches that reflect a similar level of complexity. Whatever the specific ways in which the requirements outlined in the expectations are implemented in the classroom, they must, wherever possible, be inclusive and reflect the diversity of the student population and the population of the province.

STRANDS IN THE COMPUTER STUDIES CURRICULUM

Each course in computer studies is organized into distinct but related strands. The strands are particular to each course.

The Grade 10 course is organized in three strands: A. Understanding Computers; B. Introduction to Programming; and C. Computers and Society.

Each of the Grade 11 and 12 courses has four strands. In each course, the first strand is "A. Programming Concepts and Skills", and the second is "B. Software Development". The third strand (strand C) in each of the courses focuses on the programming environment, and the Grade 12 university preparation course features Designing Modular Programs in this strand. The fourth strand (strand D) deals with the role of computers in society, and includes expectations regarding environmental concerns, the ethical use of computers, and career exploration. In this strand, the university preparation courses also explore new research in the field of information and computer science, and its implications for society.

ASSESSMENT AND EVALUATION OF STUDENT ACHIEVEMENT

BASIC CONSIDERATIONS

The primary purpose of assessment and evaluation is to improve student learning. Information gathered through assessment helps teachers to determine students' strengths and weaknesses in their achievement of the curriculum expectations in each course. This information also serves to guide teachers in adapting curriculum and instructional approaches to students' needs and in assessing the overall effectiveness of programs and classroom practices.

Assessment is the process of gathering information from a variety of sources (including assignments, demonstrations, projects, performances, and tests) that accurately reflects how well a student is achieving the curriculum expectations in a course. As part of assessment, teachers provide students with descriptive feedback that guides their efforts towards improvement. Evaluation refers to the process of judging the quality of student work on the basis of established criteria, and assigning a value to represent that quality.

Assessment and evaluation will be based on the provincial curriculum expectations and the achievement levels outlined in this document.

In order to ensure that assessment and evaluation are valid and reliable, and that they lead to the improvement of student learning, teachers must use assessment and evaluation strategies that:

- address both what students learn and how well they learn;
- are based both on the categories of knowledge and skills and on the achievement level descriptions given in the achievement chart on pages 16–17;
- are varied in nature, administered over a period of time, and designed to provide opportunities for students to demonstrate the full range of their learning;
- are appropriate for the learning activities used, the purposes of instruction, and the needs and experiences of the students;
- are fair to all students;

- accommodate the needs of students with special education needs, consistent with the strategies outlined in their Individual Education Plan;
- accommodate the needs of students who are learning the language of instruction (English or French);
- ensure that each student is given clear directions for improvement;
- promote students' ability to assess their own learning and to set specific goals;
- include the use of samples of students' work that provide evidence of their achievement;
- are communicated clearly to students and parents at the beginning of the course or the school term and at other appropriate points throughout the school year.

Evaluation of Achievement of Overall Expectations

All curriculum expectations must be accounted for in instruction, but evaluation focuses on students' achievement of the overall expectations. A student's achievement of the overall expectations is evaluated on the basis of his or her achievement of related specific expectations. The overall expectations are broad in nature, and the specific expectations define the particular content or scope of the knowledge and skills referred to in the overall expectations. Teachers will use their professional judgement to determine which specific expectations should be used to evaluate achievement of the overall expectations, and which ones will be covered in instruction and assessment (e.g., through direct observation) but not necessarily evaluated.

Levels of Achievement

The characteristics given in the achievement chart (pages 16–17) for level 3 represent the "provincial standard" for achievement of the expectations in a course. A complete picture of overall achievement at level 3 in a course in computer studies can be constructed by reading from top to bottom in the shaded column of the achievement chart, headed "70–79% (Level 3)". Parents of students achieving at level 3 can be confident that their children will be prepared for work in subsequent courses.

Level 1 identifies achievement that falls much below the provincial standard, while still reflecting a passing grade. Level 2 identifies achievement that approaches the standard. Level 4 identifies achievement that surpasses the standard. It should be noted that achievement at level 4 does not mean that the student has achieved expectations beyond those specified for a particular course. It indicates that the student has achieved all or almost all of the expectations for that course, and that he or she demonstrates the ability to use the specified knowledge and skills in more sophisticated ways than a student achieving at level 3.

THE ACHIEVEMENT CHART FOR COMPUTER STUDIES

The achievement chart that follows identifies four categories of knowledge and skills in computer studies. The achievement chart is a standard province-wide guide to be used by teachers. It enables teachers to make judgements about student work that are based on clear performance standards and on a body of evidence collected over time.

The purpose of the achievement chart is to:

- provide a common framework that encompasses all curriculum expectations for all courses outlined in this document;
- guide the development of quality assessment tasks and tools (including rubrics);
- help teachers to plan instruction for learning;
- assist teachers in providing meaningful feedback to students;
- provide various categories and criteria with which to assess and evaluate students' learning.

Categories of Knowledge and Skills

The categories, defined by clear criteria, represent four broad areas of knowledge and skills within which the subject expectations for any given course are organized. The four categories should be considered as interrelated, reflecting the wholeness and interconnectedness of learning.

The categories of knowledge and skills are described as follows:

Knowledge and Understanding. Subject-specific content acquired in each course (knowledge), and the comprehension of its meaning and significance (understanding).

Thinking. The use of critical and creative thinking skills and/or processes, as follows:

- planning skills (e.g., focusing research, gathering information, selecting strategies, organizing a project)
- processing skills (e.g., analysing, interpreting, assessing, reasoning, generating ideas, evaluating, synthesizing, seeking a variety of perspectives)
- critical/creative thinking processes (e.g., problem solving, decision making, research)

Communication. The conveying of meaning through various forms.

Application. The use of knowledge and skills to make connections within and between various contexts.

Teachers will ensure that student work is assessed and/or evaluated in a balanced manner with respect to the four categories, and that achievement of particular expectations is considered within the appropriate categories.

Criteria

Within each category in the achievement chart, criteria are provided that are subsets of the knowledge and skills that define each category. For example, in Knowledge and Understanding, the criteria are "knowledge of content (e.g., facts, technical terminology, definitions, procedures, standards)" and "understanding of content (e.g., concepts, principles, methodologies, use of tools)". The criteria identify the aspects of student performance that are assessed and/or evaluated, and serve as guides to what to look for.

Descriptors

A "descriptor" indicates the characteristic of the student's performance, with respect to a particular criterion, on which assessment or evaluation is focused. In the achievement chart, *effectiveness* is the descriptor used for each criterion in the Thinking, Communication, and Application categories. What constitutes effectiveness in any given

performance task will vary with the particular criterion being considered. Assessment of effectiveness may therefore focus on a quality such as appropriateness, clarity, accuracy, precision, logic, relevance, significance, fluency, flexibility, depth, or breadth, as appropriate for the particular criterion. For example, in the Thinking category, assessment of effectiveness might focus on the degree of relevance or depth apparent in an analysis; in the Communication category, on clarity of expression or logical organization of information and ideas; or in the Application category, on appropriateness or breadth in the making of connections. Similarly, in the Knowledge and Understanding category, assessment of knowledge might focus on accuracy, and assessment of understanding might focus on the depth of an explanation. Descriptors help teachers to focus their assessment and evaluation on specific knowledge and skills for each category and criterion, and help students to better understand exactly what is being assessed and evaluated.

Qualifiers

A specific "qualifier" is used to define each of the four levels of achievement – that is, *limited* for level 1, *some* for level 2, *considerable* for level 3, and a *high degree* or *thorough* for level 4. A qualifier is used along with a descriptor to produce a description of performance at a particular level. For example, the description of a student's performance at level 3 with respect to the first criterion in the Thinking category would be: "the student uses planning skills with *considerable* effectiveness".

The descriptions of the levels of achievement given in the chart should be used to identify the level at which the student has achieved the expectations. In all of their courses, students should be given numerous and varied opportunities to demonstrate the full extent of their achievement of the curriculum expectations across all four categories of knowledge and skills.

ACHIEVEMENT CHART: COMPUTER STUDIES, GRADES 10–12

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, technical terminology, definitions, procedures, standards)*	demonstrates limited knowledge of content	demonstrates some knowledge of content	demonstrates considerable knowledge of content	demonstrates thorough knowledge of content
Understanding of content *(e.g., concepts, principles, methodologies, use of tools)*	demonstrates limited understanding of content	demonstrates some understanding of content	demonstrates considerable understanding of content	demonstrates thorough understanding of content
Thinking – The use of critical and creative thinking skills and/or processes				
	The student:			
Use of planning skills *(e.g., focusing research, gathering information, selecting strategies, organizing a project)*	uses planning skills with limited effectiveness	uses planning skills with some effectiveness	uses planning skills with considerable effectiveness	uses planning skills with a high degree of effectiveness
Use of processing skills *(e.g., analysing, interpreting, assessing, reasoning, evaluating, integrating, synthesizing)*	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., evaluation of computer solutions, problem solving, decision making, detecting and correcting flaws, research)*	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 information *(e.g., clear expression, logical organization)* **in oral, visual, and written forms, including electronic forms** *(e.g., presentations, charts, graphs, tables, maps, models, web pages, reports)*	expresses and organizes ideas and information with limited effectiveness	expresses and organizes ideas and information with some effectiveness	expresses and organizes ideas and information with considerable effectiveness	expresses and organizes ideas and information with a high degree of effectiveness

Categories	50–59% (Level 1)	60–69% (Level 2)	70–79% (Level 3)	80–100% (Level 4)
Communication *(continued)*				
	The student:			
Communication for different audiences *(e.g., peers, computer users, company supervisor)* **and purposes** *(e.g., to inform, to persuade)* **in oral, visual, and written forms, including electronic 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 in oral, visual, and written forms, including electronic 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 *(e.g., concepts, procedures, processes, use of tools)* **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 *(e.g., choice of tools and software, ethical standards, concepts, procedures, technologies)* **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., between computer studies and personal experiences, opportunities, social and global challenges and perspectives; between subjects and disciplines)*	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

Note: A student whose achievement is below 50% at the end of a course will not obtain a credit for the course.

EVALUATION AND REPORTING OF STUDENT ACHIEVEMENT

Student achievement must be communicated formally to students and parents by means of the Provincial Report Card, Grades 9–12. The report card provides a record of the student's achievement of the curriculum expectations in every course, at particular points in the school year or semester, in the form of a percentage grade. The 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 final grade is recorded for every course, and a credit is granted and recorded for every course in which the student's grade is 50% or higher. The final grade for each course in Grades 9 to 12 will be determined as follows:

- Seventy per cent of the grade will be based on evaluations conducted throughout the course. This portion of the grade should reflect the student's most consistent level of achievement throughout the course, although special consideration should be given to more recent evidence of achievement.
- Thirty per cent of the grade will be based on a final evaluation in the form of an examination, performance, essay, and/or other method of evaluation suitable to the course content and administered towards the end of the course.

REPORTING ON DEMONSTRATED LEARNING SKILLS

The report card provides a record of the learning skills demonstrated by the student in every course, in the following five categories: Works Independently, Teamwork, Organization, Work Habits, and Initiative. The learning skills are evaluated using a four-point scale (E–Excellent, G–Good, S–Satisfactory, N–Needs Improvement). The separate evaluation and reporting of the learning skills in these five areas reflect their critical role in students' achievement of the curriculum expectations. To the extent possible, the evaluation of learning skills, apart from any that may be included as part of a curriculum expectation in a course, should not be considered in the determination of percentage grades.

SOME CONSIDERATIONS FOR PROGRAM PLANNING

Teachers who are planning a program in computer studies must take into account considerations in a number of important areas, including those discussed below.

INSTRUCTIONAL APPROACHES

Students learn best when they are engaged in learning in a variety of ways. Computer studies courses lend themselves to a wide range of approaches in that they require students to discuss issues, solve problems, plan solutions, participate in the development of solutions, conduct research, think critically, and work cooperatively. When students are engaged in active and experiential learning, they tend to retain knowledge for longer periods and to develop, acquire, and integrate key skills more completely.

Teachers need to model the skills they expect students to learn. For example, teachers should model ethical behaviour in the acquisition and use of software. They should model good program design and good coding practices in the examples and assignments they provide to students. However, in the end, students learn these skills through practice, so sufficient time to solve problems and code solutions must be provided.

Students in a computer studies class typically demonstrate diversity in the ways they learn best. It is important, therefore, that students have opportunities to learn in a variety of ways – individually, cooperatively, independently, with teacher direction, through hands-on experience, and through examples followed by practice. In computer studies, students are required to learn concepts, skills, procedures, and processes. They develop competence in these various areas with the aid of instructional and learning strategies that are suited to the particular type of learning. The approaches and strategies used in the classroom to help students meet the expectations of this curriculum should vary according to both the type of learning and the individual needs of the students.

Some of the teaching and learning strategies that are suitable to material taught in computer studies employ scaffolding. Scaffolding is an instructional approach that involves breaking down tasks so that students can concentrate on specific, manageable objectives and gradually build understanding and skill, with the aid of modelling by the teacher and ample opportunity for practice. Scaffolding provides students with a supportive structure within which to learn.

Many of the concepts taught in computer studies involve abstract thinking, which can be challenging for many students. Role playing is one approach that can help students internalize new concepts.

Learning processes that involve students in physical activity can also lead to better understanding and longer retention of concepts. The use of kinesthetic learning can be an effective way to adapt computer studies to the varied learning styles that students may demonstrate. For example, the binary numbering system could be taught by means of a game called "binary chairs", wherein eight students represent a byte of data by sitting on or standing in front of eight chairs, depending on whether they represent binary 0 (sitting) or binary 1 (standing).

Working collaboratively can enhance student learning and foster positive attitudes. An example of a collaborative approach is "pair programming", which involves two students working at a single keyboard, one as a "driver" and the other as an "observer" or "navigator". Each student has a specific role, and both students are actively involved in the task. The students periodically switch roles during the project.

The computer programming language used and the order in which concepts are taught is left to the teacher's professional judgement. Some teachers may decide to use an "objects-first" approach, while others may prefer to use structural program techniques to teach the same concepts.

Students' attitudes towards computer studies can have a significant effect on their achievement of expectations. Teaching methods and learning activities that encourage students to recognize the value and relevance of what they are learning will go a long way towards motivating them to work and learn effectively.

THE IMPORTANCE OF CURRENT EVENTS IN COMPUTER STUDIES

The study of current events and emerging technologies related to computer studies enhances both the relevance and the immediacy of the curriculum. Discussing current events and new technologies and including these topics in daily lessons stimulates student interest and curiosity and also helps students connect what they are learning in class with real-world events or situations. The study of current events needs to be thought of not as a separate topic removed from the program but as an effective instructional strategy for implementing many of the expectations found in the curriculum.

THE ROLE OF ICT IN COMPUTER STUDIES

Information and communications technologies (ICT) provide a range of tools that can significantly extend and enrich teachers' instructional strategies and support student learning. ICT tools include multimedia resources, databases, Internet websites, digital cameras, and word-processing programs. Tools such as these can help students to collect, organize, and sort the data they gather and to write, edit, and present reports on their findings. Information and communications technologies can also be used to connect students to other schools, at home and abroad, and to bring the global community into the local classroom.

Whenever appropriate, therefore, students should be encouraged to use ICT to support and communicate their learning. For example, students working individually or in groups can use computer technology and/or Internet websites to gain access to a wide range of programming resources and aids. Students can also use digital cameras and projectors to design and present the results of their research to their classmates.

Although the Internet is a powerful learning tool, there are potential risks attached to its use. All students must be made aware of issues of Internet privacy, safety, and responsible use, as well as of the potential for abuse of this technology, particularly when it is used to bully or promote hatred.

Teachers will find the various ICT tools useful in their teaching practice, both for whole-class instruction and for the design of curriculum units that contain varied approaches to learning to meet diverse student needs.

PLANNING COMPUTER STUDIES PROGRAMS FOR STUDENTS WITH SPECIAL EDUCATION NEEDS

Classroom teachers are the key educators of students who have special education needs. They have a responsibility to help all students learn, and they work collaboratively with special education resource teachers, where appropriate, to achieve this goal. *Special Education Transformation: The Report of the Co-Chairs With the Recommendations of the Working Table on Special Education, 2006* endorses a set of beliefs that should guide program planning for students with special education needs *in all disciplines*. Those beliefs are as follows:

- All students can succeed.
- Universal design[3] and differentiated instruction[4] are effective and interconnected means of meeting the learning or productivity needs of any group of students.
- Successful instructional practices are founded on evidence-based research, tempered by experience.
- Classroom teachers are key educators for a student's literacy and numeracy development.
- Each student has his or her own unique patterns of learning.
- Classroom teachers need the support of the larger community to create a learning environment that supports students with special education needs.
- Fairness is not sameness.

In any given classroom, students may demonstrate a wide range of strengths and needs. Teachers plan programs that recognize this diversity and give students performance tasks that respect their particular abilities so that all students can derive the greatest possible benefit from the teaching and learning process. The use of flexible groupings for instruction and the provision of ongoing assessment are important elements of programs that accommodate a diversity of learning needs.

3. The goal of Universal Design for Learning (UDL) is to create a learning environment that is open and accessible to all students, regardless of age, skills, or situation. Instruction based on principles of universal design is flexible and supportive, can be adjusted to meet different student needs, and enables all students to access the curriculum as fully as possible.

4. Differentiated instruction is effective instruction that shapes each student's learning experience in response to his or her particular learning preferences, interests, and readiness to learn.

In planning computer studies courses for students with special education needs, teachers should begin by examining the current achievement level of the individual student, the strengths and learning needs of the student, and the knowledge and skills that all students are expected to demonstrate at the end of the course, in order to determine which of the following options is appropriate for the student:

- no accommodations[5] or modified expectations; or
- accommodations only; or
- modified expectations, with the possibility of accommodations; or
- alternative expectations, which are not derived from the curriculum expectations for a course and which constitute alternative programs and/or courses.

If the student requires either accommodations or modified expectations, or both, the relevant information, as described in the following paragraphs, must be recorded in his or her Individual Education Plan (IEP). More detailed information about planning programs for students with special education needs, including students who require alternative programs and/or courses,[6] can be found in *The Individual Education Plan (IEP): A Resource Guide, 2004* (referred to hereafter as the *IEP Resource Guide, 2004*). For a detailed discussion of the ministry's requirements for IEPs, see *Individual Education Plans: Standards for Development, Program Planning, and Implementation, 2000* (referred to hereafter as *IEP Standards, 2000*). (Both documents are available at www.edu.gov.on.ca.)

Students Requiring Accommodations Only

Some students are able, with certain accommodations, to participate in the regular course curriculum and to demonstrate learning independently. Accommodations allow access to the course without any changes to the knowledge and skills the student is expected to demonstrate. The accommodations required to facilitate the student's learning must be identified in his or her IEP (see *IEP Standards, 2000*, page 11). A student's IEP is likely to reflect the same accommodations for many, or all, subjects or courses.

Providing accommodations to students with special education needs should be the first option considered in program planning. Instruction based on principles of universal design and differentiated instruction focuses on the provision of accommodations to meet the diverse needs of learners.

There are three types of accommodations:

- *Instructional accommodations* are changes in teaching strategies, including styles of presentation, methods of organization, or use of technology and multimedia.
- *Environmental accommodations* are changes that the student may require in the classroom and/or school environment, such as preferential seating or special lighting.
- *Assessment accommodations* are changes in assessment procedures that enable the student to demonstrate his or her learning, such as allowing additional time to complete tests or assignments or permitting oral responses to test questions (see page 29 of the *IEP Resource Guide, 2004*, for more examples).

5. "Accommodations" refers to individualized teaching and assessment strategies, human supports, and/or individualized equipment.

6. Alternative programs are identified on the IEP form by the term "alternative (ALT)".

If a student requires "accommodations only" in computer studies courses, assessment and evaluation of his or her achievement will be based on the appropriate course curriculum expectations and the achievement levels outlined in this document. The IEP box on the student's Provincial Report Card will not be checked, and no information on the provision of accommodations will be included.

Students Requiring Modified Expectations

Some students will require modified expectations, which differ from the regular course expectations. For most students, modified expectations will be based on the regular course curriculum, with changes in the number and/or complexity of the expectations. Modified expectations represent specific, realistic, observable, and measurable achievements and describe specific knowledge and/or skills that the student can demonstrate independently, given the appropriate assessment accommodations.

It is important to monitor, and to reflect clearly in the student's IEP, the extent to which expectations have been modified. As noted in Section 7.12 of the ministry's policy document *Ontario Secondary Schools, Grades 9 to 12: Program and Diploma Requirements, 1999*, the principal will determine whether achievement of the modified expectations constitutes successful completion of the course, and will decide whether the student is eligible to receive a credit for the course. This decision must be communicated to the parents and the student.

When a student is expected to achieve most of the curriculum expectations for the course, the modified expectations should identify *how the required knowledge and skills differ from those identified in the course expectations*. When modifications are so extensive that achievement of the learning expectations (knowledge, skills, and performance tasks) is not likely to result in a credit, the expectations should *specify the precise requirements or tasks on which the student's performance will be evaluated* and which will be used to generate the course mark recorded on the Provincial Report Card.

Modified expectations indicate the knowledge and/or skills the student is expected to demonstrate and have assessed in each reporting period (see *IEP Standards, 2000*, pages 10 and 11). The student's learning expectations must be reviewed in relation to the student's progress at least once every reporting period, and must be updated as necessary (see *IEP Standards, 2000*, page 11).

If a student requires modified expectations in computer studies courses, assessment and evaluation of his or her achievement will be based on the learning expectations identified in the IEP and on the achievement levels outlined in this document. If some of the student's learning expectations for a course are modified but the student is working towards a credit for the course, it is sufficient simply to check the IEP box on the Provincial Report Card. If, however, the student's learning expectations are modified to such an extent that the principal deems that a credit will not be granted for the course, the IEP box must be checked and the appropriate statement from the *Guide to the Provincial Report Card, Grades 9–12, 1999* (page 8) must be inserted. The teacher's comments should include relevant information on the student's demonstrated learning of the modified expectations, as well as next steps for the student's learning in the course.

PROGRAM CONSIDERATIONS FOR ENGLISH LANGUAGE LEARNERS

Ontario schools have some of the most multilingual student populations in the world. The first language of approximately 20 per cent of the students in Ontario's English language schools is a language other than English. Ontario's linguistic heritage includes several Aboriginal languages and many African, Asian, and European languages. It also includes some varieties of English – also referred to as dialects – that differ significantly from the English required for success in Ontario schools. Many English language learners were born in Canada and have been raised in families and communities in which languages other than English, or varieties of English that differ from the language used in the classroom, are spoken. Other English language learners arrive in Ontario as newcomers from other countries; they may have experience of highly sophisticated educational systems, or they may have come from regions where access to formal schooling was limited.

When they start school in Ontario, many of these students are entering a new linguistic and cultural environment. All teachers share in the responsibility for these students' English language development.

English language learners (students who are learning English as a second or additional language in English language schools) bring a rich diversity of background knowledge and experience to the classroom. These students' linguistic and cultural backgrounds not only support their learning in their new environment but also become a cultural asset in the classroom community. Teachers will find positive ways to incorporate this diversity into their instructional programs and into the classroom environment.

Most English language learners in Ontario schools have an age-appropriate proficiency in their first language. Although they need frequent opportunities to use English at school, there are important educational and social benefits associated with continued development of their first language while they are learning English. Teachers need to encourage parents to continue to use their own language at home in rich and varied ways as a foundation for language and literacy development in English. It is also important for teachers to find opportunities to bring students' languages into the classroom, using parents and community members as a resource.

During their first few years in Ontario schools, English language learners may receive support through one of two distinct programs from teachers who specialize in meeting their language-learning needs:

English as a Second Language (ESL) programs are for students born in Canada or newcomers whose first language is a language other than English, or is a variety of English significantly different from that used for instruction in Ontario schools.

English Literacy Development (ELD) programs are primarily for newcomers whose first language is a language other than English, or is a variety of English significantly different from that used for instruction in Ontario schools, and who arrive with significant gaps in their education. These students generally come from countries where access to education is limited or where there are limited opportunities to develop language and literacy skills in any language. Some Aboriginal students from remote communities in Ontario may also have had limited opportunities for formal schooling, and they also may benefit from ELD instruction.

In planning programs for students with linguistic backgrounds other than English, teachers need to recognize the importance of the orientation process, understanding that every learner needs to adjust to the new social environment and language in a unique way and at an individual pace. For example, students who are in an early stage of English-language acquisition may go through a "silent period" during which they closely observe the interactions and physical surroundings of their new learning environment. They may use body language rather than speech or they may use their first language until they have gained enough proficiency in English to feel confident of their interpretations and responses. Students thrive in a safe, supportive, and welcoming environment that nurtures their self-confidence while they are receiving focused literacy instruction. When they are ready to participate in paired, small-group, or whole-class activities, some students will begin by using a single word or phrase to communicate a thought, while others will speak quite fluently.

With exposure to the English language in a supportive learning environment, most young children will develop oral fluency quite quickly, making connections between concepts and skills acquired in their first language and similar concepts and skills presented in English. However, oral fluency is not a good indicator of a student's knowledge of vocabulary or sentence structure, reading comprehension, or other aspects of language proficiency that play an important role in literacy development and academic success. Research has shown that it takes five to seven years for most English language learners to catch up to their English-speaking peers in their ability to use English for academic purposes. Moreover, the older the children are when they arrive, the greater the amount of language knowledge and skills that they have to catch up on, and the more direct support they require from their teachers.

Responsibility for students' English-language development is shared by the classroom teacher, the ESL/ELD teacher (where available), and other school staff. Volunteers and peers may also be helpful in supporting English language learners in the language classroom. Teachers must adapt the instructional program in order to facilitate the success of these students in their classrooms. Appropriate adaptations include:

- modification of some or all of the subject expectations so that they are challenging but attainable for the learner at his or her present level of English proficiency, given the necessary support from the teacher;
- use of a variety of instructional strategies (e.g., extensive use of visual cues, graphic organizers, and scaffolding; previewing of textbooks; pre-teaching of key vocabulary; peer tutoring; strategic use of students' first languages);
- use of a variety of learning resources (e.g., visual material, simplified text, bilingual dictionaries, and materials that reflect cultural diversity);
- use of assessment accommodations (e.g., granting of extra time; use of oral interviews, demonstrations or visual representations, or tasks requiring completion of graphic organizers or cloze sentences instead of essay questions and other assessment tasks that depend heavily on proficiency in English).

When learning expectations in any course are modified for an English language learner (whether the student is enrolled in an ESL or ELD course or not), this information must be clearly indicated on the student's report card.

Although the degree of program adaptation required will decrease over time, students who are no longer receiving ESL or ELD support may still need some program adaptations to be successful.

For further information on supporting English language learners, refer to *The Ontario Curriculum, Grades 9–12: English as a Second Language and English Literacy Development, 2007; English Language Learners – ESL and ELD Programs and Services: Policies and Procedures for Ontario Elementary and Secondary Schools, Kindergarten to Grade 12, 2007;* and the resource guide *Many Roots, Many Voices: Supporting English Language Learners in Every Classroom, 2005.*

ANTIDISCRIMINATION EDUCATION IN COMPUTER STUDIES

The implementation of antidiscrimination principles in education influences all aspects of school life. It promotes a school climate that encourages all students to work to attain high standards, affirms the worth of all students, and helps students strengthen their sense of identity and develop a positive self-image. It encourages staff and students alike to value and show respect for diversity in the school and the wider society. It requires schools to adopt measures to provide a safe environment for learning, free from harassment, violence, and expressions of hate.

Antidiscrimination education encourages students to think critically about themselves and others in the world around them in order to promote fairness, healthy relationships, and active, responsible citizenship.

Schools have the responsibility to ensure that school-community interaction reflects the diversity in the local community and wider society. Consideration should be given to a variety of strategies for communicating and working with parents and community members from diverse groups, in order to ensure their participation in such school activities as plays, concerts, and teacher interviews. Families new to Canada, who may be unfamiliar with the Ontario school system, or parents of Aboriginal students may need special outreach and encouragement in order to feel comfortable in their interactions with the school.

When planning instructional activities for computer studies, teachers should base their decisions on the needs of students, taking into consideration the diversity of their abilities, backgrounds, interests, and learning styles. Teaching strategies, assessment and evaluation materials, and the classroom environment should be designed to value the experiences and contributions of all people.

Participation rates in computer studies tend to be higher for male students than for female students. To encourage greater participation among female students, it may be helpful to offer more projects and activities that have socially meaningful applications. For example, projects to develop assistive devices, as opposed to the more traditional activity of programming robotic arms, have proved successful in engaging the interest of female students. Similarly, projects involving graphical programming to develop animated stories, as opposed to abstract gaming activities, may hold more appeal for young women. Providing outreach programs and establishing study groups for young women may help them develop greater self-confidence in computer studies. Technology fairs and showcase events can introduce all students to a wide range of programming and

computer-related activities, such as animation and graphical programming, and may encourage an interest in programming. Offering choices from a range of instructional activities or allowing students to select their own projects can help motivate all students in a classroom by acknowledging the differences in their experiences, attitudes, and interests.

It is important to have open and frank discussions about the kind of workplace environment students are likely to encounter in the field of computer science. Inviting female and visible minority role models who have had successful careers in computer studies as guest speakers and recruiting female and visible minority senior students as mentors can enhance the interest and motivation of students for whom computer studies may be a non-traditional field. Also, exploring strategies to enable students with different learning and social styles to work effectively together can encourage participation by students whose presence will lead to a more inclusive working environment.

ENVIRONMENTAL EDUCATION AND COMPUTER STUDIES

> *Environmental education is education about the environment, for the environment, and in the environment that promotes an understanding of, rich and active experience in, and an appreciation for the dynamic interactions of:*
>
> - *The Earth's physical and biological systems*
> - *The dependency of our social and economic systems on these natural systems*
> - *The scientific and human dimensions of environmental issues*
> - *The positive and negative consequences, both intended and unintended, of the interactions between human-created and natural systems.*
>
> *Shaping Our Schools, Shaping Our Future: Environmental Education in Ontario Schools* (June 2007), p. 6

As noted in *Shaping Our Schools, Shaping Our Future*, environmental education "is the responsibility of the entire education community. It is a content area and can be taught. It is an approach to critical thinking, citizenship, and personal responsibility, and can be modelled. It is a context that can enrich and enliven education in all subject areas and offer students the opportunity to develop a deeper connection with themselves, their role in society, and their interdependence on one another and the Earth's natural systems" (p. 10).

There are many opportunities to integrate environmental education into the teaching of computer studies. In each of the computer studies courses, the expectations relating to environmental stewardship and sustainability allow students to focus on learning related to critical thinking, citizenship, and personal responsibility. Students analyse the impact of computer use on the environment. Questions about the safe handling and disposal of materials and substances used in computer studies provide students with opportunities to explore how simple human interactions with the environment can have significant consequences. Students will be expected to actively engage in developing and implementing strategies to reduce, reuse, and recycle computers, their products, and associated technologies. As well, they will research government agencies and community partners who have developed relevant opportunities to support these activities. By identifying and implementing measures to reduce the negative effects of computers on the environment, students will contribute to responsible environmental stewardship.

Programming projects can be used to support these expectations. For example, students might program a survey to assess people's knowledge of environmentally responsible strategies related to the use of computers. The program could be designed to calculate the respondent's "environmental awareness" grade and suggest additional strategies, or it could be designed to provide feedback for each survey question. Students could also design surveys to assess the use of environmentally responsible practices in the classroom.

Environmental education can also be integrated into the design of other programming projects, such as simulations that model healthy ecosystems (showing the balance between plants and animals in an enclosed system); or the consequences of an environmental catastrophe such as an oil spill on a coastline (including the speed and depth of the oil spread and the impact of the oil on the area affected); or the social costs and benefits of designing energy-efficient buildings. The dynamic relationships resulting from human interaction with the environment provide a rich context for developing authentic learning activities within computer studies courses.

LITERACY, MATHEMATICAL LITERACY, AND INQUIRY/RESEARCH SKILLS

Literacy, mathematical literacy, and inquiry/research skills are critical to students' success in all subjects of the curriculum and in all areas of their lives.

Many of the activities and tasks that students undertake in the computer studies curriculum involve literacy skills relating to oral, written, and visual communication. For example, students use language to describe their observations, to describe their critical analyses in both informal and formal contexts, and to present their findings in presentations and reports in oral, written, graphic, and multimedia forms. Computer studies also requires the use and understanding of specialized terminology. In all computer studies courses, students are expected to use appropriate and correct terminology, and are encouraged to use language with care and precision in order to communicate effectively.

The computer studies program also builds on, reinforces, and enhances mathematical literacy. For example, clear, concise communication often involves the use of diagrams, tables, and graphs, and many components of the computer studies curriculum emphasize students' ability to interpret and use symbols and charts.

Inquiry is at the heart of learning in all subject areas. In computer studies courses, students are encouraged to develop their ability to ask questions and to explore a variety of possible answers to those questions. As they advance through the grades, they acquire the skills to locate relevant information from a variety of sources, such as books, periodicals, dictionaries, encyclopedias, interviews, videos, and the Internet. The questioning they practised in the early grades becomes more sophisticated as they learn that all sources of information have a particular point of view and that the recipient of the information has a responsibility to evaluate it, determine its validity and relevance, and use it in appropriate ways. The ability to locate, question, and evaluate information allows a student to become an independent, lifelong learner.

THE ONTARIO SKILLS PASSPORT AND ESSENTIAL SKILLS

Teachers planning programs in computer studies need to be aware of the purpose and benefits of the Ontario Skills Passport (OSP). The OSP is a bilingual web-based resource that enhances the relevance of classroom learning for students and strengthens school–

work connections. The OSP provides clear descriptions of Essential Skills such as Reading Text, Writing, Computer Use, Measurement and Calculation, and Problem Solving and includes an extensive database of occupation-specific workplace tasks that illustrate how workers use these skills on the job. The Essential Skills are transferable, in that they are used in virtually all occupations. The OSP also includes descriptions of important work habits, such as working safely, being reliable, and providing excellent customer service. The OSP is designed to help employers assess and record students' demonstration of these skills and work habits during their cooperative education placements. Students can use the OSP to assess, practise, and build their Essential Skills and work habits and transfer them to a job or further education or training.

The skills described in the OSP are the Essential Skills that the Government of Canada and other national and international agencies have identified and validated, through extensive research, as the skills needed for work, learning, and life. These Essential Skills provide the foundation for learning all other skills and enable people to evolve with their jobs and adapt to workplace change. For further information on the OSP and the Essential Skills, visit http://skills.edu.gov.on.ca.

CAREER EDUCATION

Ongoing discoveries and innovations coupled with rapidly evolving technologies have resulted in an exciting environment in which creativity and innovation thrive, bringing about new career opportunities. Today's employers seek candidates with strong technical skills, critical-thinking and problem-solving skills, and the ability to work cooperatively in a team, traits that are developed through participation in computer studies courses. Computer studies courses enable students to develop, for example, problem-solving skills, design skills, technical knowledge and skills, and the ability to conduct research, present results, and work on projects both independently and in a team environment.

COOPERATIVE EDUCATION AND OTHER FORMS OF EXPERIENTIAL LEARNING

Cooperative education and other forms of experiential learning, such as job shadowing, field trips, and work experience, enable students to apply the skills they have developed in the classroom to real-life activities in the community and in the world of business and public service. Cooperative education and other workplace experiences also help to broaden students' knowledge of employment opportunities in a wide range of fields, including computer programming, database analysis, computer science, education, computer engineering, software engineering, information technology, and game development. In addition, students develop their understanding of workplace practices, certifications, and the nature of employer–employee relationships. Teachers of computer studies can support their students' learning by maintaining links with community-based businesses to ensure that students have access to hands-on experiences that will reinforce the knowledge and skills gained in school.

Students who choose a computer studies course as the related course for two cooperative education credits are able, through this packaged program, to meet the OSSD compulsory credit requirements for groups 1, 2, and 3.

Health and safety issues must be addressed when learning involves cooperative education and other workplace experiences. Teachers who provide support for students in workplace learning placements need to assess placements for safety and ensure that

students understand the importance of issues relating to health and safety in the workplace. Before taking part in workplace learning experiences, students must acquire the knowledge and skills needed for safe participation. Students must understand their rights to privacy and confidentiality as outlined in the Freedom of Information and Protection of Privacy Act. They have the right to function in an environment free from abuse and harassment, and they need to be aware of harassment and abuse issues in establishing boundaries for their own personal safety. They should be informed about school and community resources and school policies and reporting procedures with respect to all forms of abuse and harassment.

Policy/Program Memorandum No. 76A, "Workplace Safety and Insurance Coverage for Students in Work Education Programs" (September 2000), outlines procedures for ensuring the provision of Health and Safety Insurance Board coverage for students who are at least 14 years of age and are on placements of more than one day. (A one-day job-shadowing or job-twinning experience is treated as a field trip.) Teachers should also be aware of the minimum age requirements outlined in the Occupational Health and Safety Act for persons to be in or to be working in specific workplace settings. All cooperative education and other workplace experiences will be provided in accordance with the ministry's policy document *Cooperative Education and Other Forms of Experiential Learning: Policies and Procedures for Ontario Secondary Schools, 2000.*

PLANNING PROGRAM PATHWAYS AND PROGRAMS LEADING TO A SPECIALIST HIGH SKILLS MAJOR

Computer studies courses are well suited for inclusion in some programs leading to a Specialist High Skills Major (SHSM) or in programs designed to provide pathways to particular apprenticeship or workplace destinations. In some SHSM programs, computer studies courses can be bundled with other courses to provide the academic knowledge and skills important to particular industry sectors and required for success in the workplace and postsecondary education, including apprenticeship. Computer studies courses may also be combined with cooperative education credits to provide the workplace experience required for SHSM programs and for various program pathways to apprenticeship and workplace destinations. (SHSM programs would also include sector-specific learning opportunities offered by employers, skills-training centres, colleges, and community organizations.)

HEALTH AND SAFETY IN COMPUTER STUDIES

The major health and safety concerns associated with computer use are musculoskeletal injuries (including repetitive strain injuries) and eye strain. Teachers should not only ensure that workstations are ergonomically arranged but also encourage students to maintain good posture and to take regular breaks to stand and stretch. It is also important to inform students of the mental and emotional health risks associated with social isolation – a familiar condition among heavy computer users.

Various kinds of health and safety issues can arise when learning involves field trips. Out-of-school field trips can provide an exciting and authentic dimension to students' learning experiences. They also take the teacher and students out of the predictable classroom environment and into unfamiliar settings. Teachers must preview and plan these activities carefully to protect students' health and safety.

COURSES

Introduction to Computer Studies, Grade 10

Open — ICS2O

This course introduces students to computer programming. Students will plan and write simple computer programs by applying fundamental programming concepts, and learn to create clear and maintainable internal documentation. They will also learn to manage a computer by studying hardware configurations, software selection, operating system functions, networking, and safe computing practices. Students will also investigate the social impact of computer technologies, and develop an understanding of environmental and ethical issues related to the use of computers.

Prerequisite: None

A. UNDERSTANDING COMPUTERS

OVERALL EXPECTATIONS

By the end of this course, students will:

A1. describe the functions of different types of hardware components, and assess the hardware needs of users;

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;

A4. demonstrate an understanding of home computer networking concepts;

A5. explain the importance of software updates and system maintenance to manage the performance and increase the security of a computer.

SPECIFIC EXPECTATIONS

A1. Hardware Components

By the end of this course, students will:

A1.1 use correct terminology to describe computer hardware *(e.g., USB, FSB, IEEE 1394 interface)*, speed measurements *(e.g., megahertz)*, and size measurements *(e.g., megabytes, gigabytes)*;

A1.2 describe the functions of the internal components of a computer *(e.g., CPU, RAM, ROM, cache, hard drive, motherboard, power supply, video card, sound card)*;

A1.3 describe the functions of common computer peripheral devices *(e.g., printer, monitor, scanner, keyboard, mouse, speakers, USB flash drive)*;

A1.4 assess user computing needs and select appropriate hardware components 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)*.

A2. Software Products

By the end of this course, students will:

A2.1 explain the difference between software used for applications *(e.g., word processor, spreadsheet, email client)*, programming *(e.g., an integrated development environment)*, and systems *(e.g., operating system tools such as a registry editor and a defragmenting tool)*;

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

By the end of this course, students will:

A3.1 describe operating system functions that meet various user needs *(e.g., running applications, organizing files, managing users, configuring peripherals)*;

A3.2 use file management techniques to organize and manage files *(e.g., copy, move, delete, rename files; create shortcut)*;

A3.3 use general keyboard shortcuts to perform common tasks *(e.g., cut, copy, paste, print, print window, print screen)*;

A3.4 describe the features and limitations of various operating systems.

A4. Home Computer Networking

By the end of this course, students will:

A4.1 identify various networking applications and protocols *(e.g., VoIP, streaming media, FTP, email, instant messaging)*;

A4.2 describe the features and functions of wired and wireless networking hardware *(e.g., NICs, routers, hubs, cables, modems)*;

A4.3 demonstrate an understanding of various methods for sharing network resources *(e.g., shared file access, shared printer access, Internet access)*.

A5. Maintenance and Security

By the end of this course, students will:

A5.1 describe different types of malware *(e.g., viruses, Trojan horses, worms, spyware, adware, malevolent macros)* and common signs of an intrusion, and explain how to prevent malware attacks;

A5.2 explain the importance of maintaining software updates *(e.g., operating system updates, application software updates, virus definitions)* to increase computer security and maintain hardware and software compatibility;

A5.3 explain the importance of preventive maintenance *(e.g., defragmenting a hard drive, deleting unused software and data files)* to manage computer performance.

B. INTRODUCTION TO PROGRAMMING

OVERALL EXPECTATIONS

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

B1. Programming Concepts

By the end of this course, students will:

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

By the end of this course, students will:

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)*;

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. Code Maintenance

By the end of this course, students will:

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;

B3.4 demonstrate the ability to validate a computer program using test cases.

C. COMPUTERS AND SOCIETY

OVERALL EXPECTATIONS

By the end of this course, students will:

C1. describe key aspects of the impact of computers and related technologies on society;
C2. describe computer use policies that promote environmental stewardship and sustainability;
C3. describe legal and ethical issues related to the use of computing devices;
C4. describe postsecondary education and career prospects related to computer studies.

SPECIFIC EXPECTATIONS

C1. Social Impact

By the end of this course, students will:

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)*.

C2. Environmental Stewardship and Sustainability

By the end of this course, students will:

C2.1 describe the negative effects of computers and computer use on the environment *(e.g., chemicals from electronic waste dumped in landfills – domestic or overseas – leaching into soil and groundwater; unnecessary use of paper; heavy power consumption)* and on human health *(e.g., effects of exposure to radiation, musculoskeletal disorders, eye strain, mental health and behavioural problems created or exacerbated by social isolation)*;

C2.2 identify measures that help reduce the negative effects of computers on the environment *(e.g., lab regulations, school policies, corporate policies, provincial policies, paperless workplaces)* and on human health *(e.g., ergonomic standards)*;

C2.3 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; interpretation of large amounts of environmental data; management of natural resources; programmable temperature control to reduce energy consumption)*;

C2.4 describe, on the basis of research, how and where recycled electronic waste is processed, and identify local companies and institutions that offer such services.

C3. Ethical Issues

By the end of this course, students will:

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

By the end of this course, students will:

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 interested 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.

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.

Introduction to Computer Programming, Grade 11

College Preparation — ICS3C

This course introduces students to computer programming concepts and practices. Students will write and test computer programs, using various problem-solving strategies. They will learn the fundamentals of program design and apply a software development life-cycle model to a software development project. Students will also learn about computer environments and systems, and explore environmental issues related to computers, safe computing practices, emerging technologies, and post-secondary opportunities 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 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.

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 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.

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 demonstrate the ability to write algorithms with nested structures.

A3. Code Maintenance

By the end of this course, students will:

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)*;

A3.5 demonstrate the ability to validate a program using 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;

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 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., 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

By the end of this course, students will:

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)*.

B3. Designing Simple Algorithms

By the end of this course, students will:

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;

B3.3 design algorithms to detect, intercept, and handle run-time errors *(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. demonstrate an understanding of the functions of different types of computer components;

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

C3. use a software development environment to write and run computer programs.

SPECIFIC EXPECTATIONS

C1. Computer Components

By the end of this course, students will:

C1.1 describe the functions and features of the internal components of a computer *(e.g., CPU, RAM, ROM, cache, hard drive, motherboard, power supply, video card, sound card)*;

C1.2 use correct terminology to describe computer features and specifications *(e.g., processor type, bus speed, storage capacity, amount of memory)*;

C1.3 describe the functions and features of common computer peripheral devices *(e.g., printer, monitor, scanner, keyboard, mouse, speakers, USB flash drive)*;

C1.4 compare and contrast common ISP services *(e.g., DSL, cable, dial-up, regional Wi-Fi)* and home networking hardware *(e.g., NICs, routers, hardware used for wired and wireless connections)*.

C2. File Maintenance

By the end of this course, students will:

C2.1 use an operating system to logically organize computer files for easy retrieval, backup, and recovery;

C2.2 use standard backup procedures to back up user files.

C3. The Software Development Environment

By the end of this course, students will:

C3.1 describe the functions and features of a software development environment and use it to write and run a computer program;

C3.2 describe the differences between applications, programming languages, and operating systems;

C3.3 use Help documentation as a guide to designing and writing programs.

D. COMPUTERS AND SOCIETY

OVERALL EXPECTATIONS

By the end of this course, students will:

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

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.

SPECIFIC EXPECTATIONS

D1. Environmental Stewardship and Sustainability

By the end of this course, students will:

D1.1 describe negative effects of computer use on the environment *(e.g., creation of waste, unnecessary printing of emails, heavy power consumption)* and on human health *(e.g., exposure to radiation, musculoskeletal disorders, eye strain, 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 policies, provincial policies, paperless workplaces, 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; interpretation of large amounts of environmental data; management of natural resources; programmable temperature control to reduce energy consumption)*;

D1.4 identify government agencies and community partners that provide environmental stewardship opportunities *(e.g., local community recycling centres, private companies that refurbish computers, printer cartridge recycling programs)*.

D2. Safe Computing

By the end of this course, students will:

D2.1 explain the need for an acceptable-use policy for using computers at school and at work;

D2.2 describe and use appropriate strategies to avoid potential health and safety problems associated with computer use *(e.g., musculoskeletal disorders, eye strain)*;

D2.3 describe procedures to safeguard data and programs from malware *(e.g., viruses, spyware, adware)*.

D3. Emerging Technologies

By the end of this course, students will:

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. Postsecondary Opportunities

By the end of this course, students will:

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)*;

D4.2 identify opportunities for experiential learning *(e.g., co-op programs, job shadowing, career fairs)* related to computer science;

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)*;

D4.4 identify groups and programs that are available to support students who are interested 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)*;

D4.5 describe the Essential Skills and work habits that are important for success in computer studies, as identified in the Ontario Skills Passport.

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.

Computer Programming, Grade 12

College Preparation — ICS4C

This course further develops students' computer programming skills. Students will learn object-oriented programming concepts, create object-oriented software solutions, and design graphical user interfaces. Student teams will plan and carry out a software development project using industry-standard programming tools and proper project management techniques. Students will also investigate ethical issues in computing and expand their understanding of environmental issues, emerging technologies, and computer-related careers.

Prerequisite: Introduction to Computer Programming, Grade 11, College Preparation

A. PROGRAMMING CONCEPTS AND SKILLS

OVERALL EXPECTATIONS

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

A1. Data Structures

By the end of this course, students will:

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.

A2. Using Standard Algorithms

By the end of this course, students will:

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)*;

A2.2 demonstrate the ability to read from, and write to, an external file *(e.g., sequential file, database, XML file, relational database via SQL)*;

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., string.indexOf ("cool"), Arrays.sort(intArray))* in a program.

A3. Object-oriented Programming

By the end of this course, students will:

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.

A4. Code Maintenance

By the end of this course, students will:

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

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

B1. Designing Standard Algorithms

By the end of this course, students will:

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

By the end of this course, students will:

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

By the end of this course, students will:

B3.1 design graphical user interfaces that contain common controls *(e.g., buttons, labels, text boxes)*;

B3.2 design a user-friendly graphical user interface 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

By the end of this course, students will:

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

By the end of this course, students will:

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.

SPECIFIC EXPECTATIONS

C1. Project Management Tools

By the end of this course, students will:

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

By the end of this course, students will:

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

By the end of this course, students will:

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

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.

SPECIFIC EXPECTATIONS

D1. Environmental Stewardship and Sustainability

By the end of this course, students will:

D1.1 outline and apply 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 describe governmental and community initiatives promoting environmental stewardship and 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 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

By the end of this course, students will:

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

By the end of this course, students will:

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.

The Ministry of Education wishes to acknowledge the contributions of the many individuals, groups, and organizations that participated in the development and refinement of this curriculum policy document.

Printed on recycled paper

08-027
ISBN 978-1-4249-8102-1 (Print)
ISBN 978-1-4249-8103-8 (PDF)
ISBN 978-1-4249-8104-5 (TXT)