



Computer Science 20, 30: Curriculum Guidelines for the Secondary Level



Computer Science 20, 30 Curriculum Guidelines for the Secondary Level

Saskatchewan Education
1999
ISBN: 1-894116-22-4

Acknowledgements

Saskatchewan Education acknowledges the following members of the ad hoc Reference Committee for their contributions:

Ed Adolph
Teacher
Regina S.D. #4

Len Proctor
College of Education
University of Saskatchewan

Karl Germann
Graduate student
University of Regina

Ron Provali
Teacher/Principal
Potashville S.D. #80

Jim Greer
Computer Science Department
University of Saskatchewan

Don Roszell
Manager, Information Services
Saskatoon S.D. #13

Lori Howard
Teacher
Saskatchewan Rivers S.D. #119

Helen Sukovieff
Teacher
Regina S.D. #4

Daryl Koroluk
Teacher
Saskatoon S.D. #13

Judi Thomson
Computer Science Department
University of Saskatchewan

Jerome Linnell
Teacher
Estevan S.D. #95

Ron Longhurst
Consultant
Saskatchewan Rivers S.D. #119

Brien Macguire
Computer Science Department
University of Regina

Saskatchewan Education wishes to thank many others who contributed to the development of these guidelines:

- Lynn Anderson, seconded/contracted developer/writer
- various Saskatchewan Education personnel
- field test/pilot teachers
- other field personnel.

This document was completed under the direction of the Science and Technology Unit, Curriculum and Instruction Branch, Saskatchewan Education.

Table of Contents

Acknowledgements	i
Foreword	1
Introduction	2
Philosophy, Aim, and Goals	2
Foundational Objectives.....	2
Using the Curriculum Guidelines	3
Course Overview	3
Scope and Sequence	3
Guide To Using Resource Materials	3
Core Curriculum Components and Initiatives	3
An Overview of Computer Science 20 and 30	4
Common Essential Learnings	5
Adaptive Dimension	5
Resource-based Learning	7
Gender Equity	8
Aboriginal Curriculum Content and Perspectives	8
Instructional Approaches	9
Assessment and Evaluation	10
Why Consider Assessment and Evaluation?.....	10
Student Evaluation	10
Program Evaluation	10
Curriculum Evaluation	11
Teacher Self-Evaluation.....	11
Clarification of Terms.....	11
Phases of the Evaluation Process.....	12
Principles of Evaluation	13
Student Assessment in Computer Science.....	13
Record Keeping	15
Assigning Grades	15
Informing Students and Parents or Guardians about Evaluation	16
Conclusion	16
Program Considerations	16
Problem Solving	17
Integration.....	17
Assignments	17
References	18
Computer Science 20	19
Outline of Foundational and Learning Objectives	19
Unit 1: Software and Hardware	21
Unit 2: Problem Solving.....	23
Unit 3: Fundamentals of Programming and Design.....	24
Unit 4: Experience with Programming and Design.....	26
Unit 5: Networks (Optional).....	30
Unit 6: Careers Related to Computer Science	32
Unit 7: Research Topics.....	33
List of Topics	34

Computer Science 30	36
Outline of Foundational and Learning Objectives	36
Unit 1: Software and Hardware - Advanced	38
Unit 2: Problem Solving - Advanced	39
Unit 3: Problem Solving and Programming.....	41
Unit 4: Experience with Programming and Design - Advanced.....	44
Unit 5: Number Systems and Codes (Optional)	47
Unit 6: Impact of Information Technology	49
Unit 7: Programming for Applications	52
Unit 8: Internet and Multimedia (Optional)	53
Unit 9: Research Topics	54
List of Topics	55
Appendix A: Suggested Languages	60

Foreword

Much of the foundation for curriculum renewal in Saskatchewan schools is based on *Directions* (1984) and subsequent decisions. The excitement surrounding the recommendations for Core Curriculum developments will continue to build as curricula are designed and implemented to prepare students for and in the 21st century.

Computer science, is an elective. It incorporates the Common Essential Learnings and the Adaptive Dimension. In addition, other Core Curriculum initiatives such as Gender Equity, Aboriginal content and perspectives, and Resource-based Learning are also addressed.

As we strive to achieve the goals of computer science education in Saskatchewan schools, much collaboration and cooperation among individuals and groups will be required. Computer science teachers are a key part of the process.

Introduction

Philosophy, Aim, and Goals

The philosophy of computer science education in Saskatchewan is reflected in the program aim and goals. In addition, the philosophy is closely related to the concept of Core Curriculum based on the philosophy of *Directions* (1984) and the **Goals of Education for Saskatchewan**.

The main aim of the computer science program is to prepare literate individuals who value computer science and appreciate its role in society. The computer science program is intended to stimulate a spirit of inquiry by developing a variety of problem-solving and decision-making skills and abilities.

The following curriculum goals are intended to provide students with the computer science preparation essential to:

- build students' skills in and enjoyment of critical thinking, problem solving, and their ability to handle unfamiliar situations. Many objectives related to Critical and Creative Thinking, as a Common Essential Learning (CEL), will be found in these courses;
- enhance students' technical language skills, with emphasis on interpreting and writing. These skills compliment certain aspects of Language Arts, but focus on the CELs of Communication and Numeracy;
- encourage positive attitudes towards technology, and promote flexibility in students' responses to technological change. Many aspects of Technological Literacy have a high profile in computer science;
- provide opportunities for students to make creative use of computers. Creativity and design are important concepts and processes in computer science;
- ensure an appropriate foundation of skills and attitudes in students:
 - ◇ planning to continue their formal education at the post-secondary level
 - ◇ planning to seek jobs involving information technology. Approaches to career development and awareness, transition-to-work considerations,

Independent Learning, and Personal and Social Values and Skills need to be supported within computer science; and,

- provide sufficient experience that students will become informed consumers of computer services, hardware, and software, and will be aware of the limitations of computer systems. With computers and their influences on society changing rapidly, computer science students must be able to make critical decisions about their and society's use of computers.

Foundational Objectives

The Foundational Objectives describe the most important understandings and abilities that should be developed over the course of a unit or a year. They define the types of learning related to the goals of a program that may be expected at a particular developmental level or year. They ensure a consistency between the goals of the program and specific curriculum planning. They provide guidance to teachers in unit and yearly planning and should be within the range of abilities of the majority of students.

The Foundational Objectives form the basis for curriculum evaluation.

The Foundational Objectives are listed at the start of the 20 and 30 level course sections and specified in the Units. Common Essential Learnings Foundational Objectives to be emphasized are specified for each Unit, also.

Coding of CELs:

COM	Communication
CCT	Critical and Creative Thinking
IL	Independent Learning
NUM	Numeracy
PSVS	Personal and Social Values and Skills
TL	Technological Literacy

Using the Curriculum Guidelines

Course Overview

See the table on the next page. The Computer Science objectives, for reasons of clarity and emphasis, are divided into various topics. These topics include: problem solving, the microcomputer itself (hardware and operating systems), programming, networks, computers and society. Topics for integration include: careers, multimedia, and projects. Many research studies indicate that instruction in computer science should regularly demonstrate its applicability to real life. Therefore, even though the student learning objectives are detailed separately, they are intended to be taught in an integrated fashion. As problem solving is an integral component of computer science, it is to be incorporated throughout the program.

Any single resource will not cover all of the concepts and skills of these curriculum guidelines. Instead, a variety of resources should be employed. From these, select activities and content that coincide with student learning styles, individual teaching styles, and the philosophy of the curriculum. As well, other community resources should be used.

The development and sequencing of the main concepts and ideas should follow a logical progression to show the relationships between the concepts. Teachers should not restrict themselves to using just one instructional strategy when teaching a concept. When choosing among the teaching strategies available, teachers might consider the intellectual aptitude of their students, what they already know about the concept, the nature of the concept, its significance in the structure of the other computer science concepts, and the level of performance expected.

Frequently, there are several appropriate ways to accomplish any particular learning objective. Students should be encouraged to use a variety of approaches and to explain their methods. Through discussions with the teachers and with other students, they gain a better insight into computer science concepts and become better able to use computer science terminology correctly.

Outlines of the topics are presented at the beginning of the 20 and 30 level courses

Scope and Sequence

The topical scope and sequence gives the reader an overview of the specific objectives as they occur in each of the courses. Depending on the amount of integration, flexibility with the sequence is permitted. Core topics and timelines have been indicated; the choice of optional topics and timelines is a local decision within the 100 instructional hours per credit for each course.

Guide To Using Resource Materials

As was indicated earlier, no single resource matches the Computer Science curriculum. To facilitate a resource-based approach, the use of a variety of resources instead of a single textbook is highly recommended. Some teachers may wish to have their students work in groups of two or three. Multimedia and online resources are recommended. In any case, the approach should coincide with student learning styles and individual teaching styles.

A Resource-based Learning approach requires long-term planning and coordination within a school or school division. In-school administrators, the teacher-librarian, and others need to take an active role to assist with this planning.

Instructional approaches that emphasize group work and develop independent learning abilities make it possible to utilize limited resources in a productive way.

Core Curriculum Components and Initiatives

Core Curriculum: An Information Bulletin for Administrators (Saskatchewan Education, 1997) defines the Core Curriculum as including the seven Required Areas of Study, the six Common Essential Learnings, the Adaptive Dimension, and Locally-Determined Options. Computer Science is a science elective.

An Overview of Computer Science 20 and 30

Computer Science 20

Unit	Title	Suggested Time (hours)
1	Software and Hardware	5-10
2	Problem Solving	5-10
3	Fundamentals of Programming and Design	5-10
4	Experience with Programming and Design	50-65
5	Networks (Optional)	5
6	Careers Related to Computer Science	5
7	Research Topics	7-10
		Minimum 100 hours

Computer Science 30

Unit	Title	Suggested Time (hours)
1	Software and Hardware -- Advanced	2-3
2	Problem Solving -- Advanced	3-4
3	Problem Solving and Programming	3-4
4	Experience with Programming and Design -- Advanced	50-65
5	Number Systems and Codes (Optional)	5-8
6	Impact of Information Technology	3-5
7	Programming for Applications	3-4
8	Internet and Multimedia (Optional)	4-6
9	Research Topics	7-10
		Minimum 100 hours

Common Essential Learnings

Understanding the Common Essential Learnings: A Handbook for Teachers (1988) defines and expands on an understanding of these essential learnings.

Computer Science offers many opportunities for incorporating the Common Essential Learnings (CELs) into instruction. The purpose of this incorporation is to help students better understand the subject matter under study and to prepare students for future learning both inside and outside of the K-12 educational system. The decision to focus on particular CELs within a lesson is guided by the needs and abilities of individual students and by the particular demands of computer science. Throughout a topic, it is intended that each Common Essential Learning will be developed to the fullest extent possible.

It is important to incorporate CELs in an authentic manner. For example, some areas of Computer Science may offer many opportunities to develop the understandings, values, skills, and processes related to a number of Common Essential Learnings. The development of a particular CEL, however, may be limited by the nature of the topic.

It is intended that Common Essential Learnings be developed and evaluated within subject areas. Therefore, Foundational Objectives map to the CELs. Because Common Essential Learnings are not necessarily separate and discrete categories, it is anticipated that working toward the achievement of one Foundational Objective may contribute to the development of other CELs. For example, many of the processes, skills, understandings, and abilities required for the CELs of Communication, Numeracy, and Critical and Creative Thinking are also needed for the development of Technological Literacy.

Incorporating Common Essential Learnings into instruction has implications for the assessment of student learning. A topic that has focused on developing the CELs of Communication and Critical and Creative Thinking should also reflect this focus when assessing student learning. Assessment should allow students to demonstrate their understanding of the important concepts in the topic and how these concepts are related to each other or to previous learning. Questions can be structured so that evidence or reasons must accompany student explanations. If students are encouraged to think critically and creatively

throughout a topic, then the assessment for the topic should also require students to think critically and creatively.

It is anticipated that teachers will build from the suggestions in these curriculum guidelines and from their personal reflections in order to incorporate Common Essential Learnings more fully into computer science. For example, involving students in groups to solve realistic problems helps to develop Personal and Social Values and Skills. Similarly, realistic problems provide a medium to promote the important aspects of human communication: listening, speaking, reading, and writing. Additionally, Critical and Creative Thinking can be developed in the computer science program by providing students with an opportunity to ask "what if" questions. Numeracy is naturally developed throughout, especially in the use of programming languages. Having students actively use the computer as a problem-solving tool and apply computer science principles to organize data and technical information develops their awareness of the importance of information technology in an ever-changing world. Independent Learning is fostered by encouraging students to investigate the applications, history, and further study of computer science. In creating such opportunities and experiences, students will become capable, self-reliant, self-motivated, and life-long learners.

Adaptive Dimension

The adaptation of instruction to meet learner needs is an expectation inherent in the **Goals of Education** and is an essential ingredient of any consideration of Instructional Approaches. Like the Common Essential Learnings, the Adaptive Dimension is a component of Core Curriculum and permeates all curriculum and instruction. The Adaptive Dimension is defined as:

... the concept of making adjustments in approved educational programs to accommodate diversity in student learning needs. It includes those practices the teacher undertakes to make curriculum, instruction, and the learning environment meaningful and appropriate for each student. (*The Adaptive Dimension in Core Curriculum*, Saskatchewan Education 1992, p. 1)

The essence of the Adaptive Dimension rests in the phrase "seeking other ways". Offering students alternative access to and expression of knowledge facilitates their participation in learning. Just as physical environments can be made more accessible through modifications such

as ramps or wider doorways, learning environments can be made more accessible through a modification of setting, method, or material. The Adaptive Dimension is used to:

- maximize student independence
- maximize generalization and transfer
- lessen discrepancies between achievement and ability
- promote a love of learning
- promote a positive self-image and feeling of belonging
- promote confidence
- promote a willingness to become involved in learning
- facilitate integration.

These purposes address a primary function of the school -- that of helping students to maximize their potentials as independent learners.

The cues that some students' needs are not being adequately met come from a variety of sources. They may come to the perceptive teacher as a result of monitoring for comprehension during a test or from a student need or background deficiency that has been recognized for several years. A student's demonstrated knowledge of, or interest in, a particular topic may indicate that enrichment is appropriate. The adaptation required may vary from presenting the same content through a slightly different instructional method, to modifying the content because of a known information background deficit, or to establishing an individual or small group enrichment activity. The duration of the adaptation may range from five minutes of individual assistance, to providing opportunities for some students to expand their knowledge and extend their understandings by working independently or with a group of similar students. The assessment of the need may be handled adequately by the classroom teacher, or may require the expertise of other support specialists such as the school's resource teacher, other system-based personnel, or community resources.

Adaptations of teaching methodologies, curriculum organization, timetabling, or the appropriate use of technology assist students, who may find learning difficult, and others who are not sufficiently challenged, to become active participants in their learning. Often students will come to a Computer Science class with a strong background in some aspects of the course. Keeping these students interested while helping others who operate at a lower level is an

important task for teachers. It is important that these students are challenged by their work in Computer Science and see that their advanced knowledge and expertise is acknowledged and appreciated in the class. Teachers should also be alert to deficiencies in these students' knowledge; often self-taught computer whizzes lack in some areas necessary in a well rounded computer scientist.

Some general guidelines for adaptation follow:

- alter the method of instruction to meet the needs of individuals
- alter the pace of the lesson to ensure that students understand the concept being presented or are being challenged by the presentation. One of the most basic adaptations that can be made to assist students is to give them sufficient time to explore, create, question, and experience as they learn
- allow more than one way to accomplish a task
- monitor the use of vocabulary. It is possible to use advanced and simple vocabulary in the same lesson by incorporating both levels in a sentence: "Show me functions that link or string things together." This helps to satisfy the requirements of some students expand the vocabulary of others, and make the lesson meaningful to others
- alter the manner in which the student is required to respond to the teacher and/or to the instructional approach
- alter the setting so that the student may benefit more fully from the instruction
- accommodate a variety of learning modes (visual, auditory, kinesthetic)
- use resources that maximize learning
- alter the complexity of programming projects to match the abilities of the students
- use interactive techniques that allow for close observation of the student's progress
- involve students in decisions regarding their own learning
- use assessment techniques that are matched to the instructional adaptations that have been made for the students.

The Adaptive Dimension includes all practices the teacher employs to make learning meaningful and appropriate for each student. Because the Adaptive Dimension permeates all teaching practice, professional decisions become the critical factor. This curriculum guideline encourages such flexibility and decision-making. Further information can be found in *The Adaptive Dimension in Core Curriculum* (1992).

Resource-based Learning

Personnel, collections, facilities, and budgets in Saskatchewan school libraries vary a great deal, and the quality of school library programs and services is therefore not consistent. Possibilities for resource-based instruction are related to the level of administrative and staff commitment to developing well-staffed and well-equipped school libraries.

Resource-based teaching and learning is a means by which teachers can greatly assist the development of attitudes and abilities for independent life-long learning. Resource-based Learning is student-centred. It offers students opportunities to choose, to explore, and to discover. Students who are encouraged to think critically in an environment rich in resources are well on their way to becoming autonomous learners.

It is important for the computer science teacher to cooperate with library staff to integrate non-print, human, and print resources with classroom assignments. The classroom teacher plans in advance with library staff and uses the library resource centre as an extension of the classroom and a place for active learning. The teacher-librarian, if available, could assist with planning assignments, integrating appropriate resources, and teaching students the processes needed to find, use, and present information.

The library resource centre staff may support the computer science curriculum by:

- exhibiting curiosity; modelling open-ended investigations, and problem-solving approaches
- demonstrating use of electronic tools to access sources that support computer science interests
- organizing and circulating print and non-print resources that support the computer science curriculum. Resources might include lists of World Wide Web sites, electronic databases, commercial games, videos, filmstrips and films, instructional software, newspapers and magazines, reference books, a resource file of speakers and presenters in the community with computer science career experience
- assisting the computer science teacher to set up learning stations in the classroom or library using library resources
- cooperating with the computer science teacher to teach students methods of library

-
- organization including computerized systems and practical uses of indexing of all kinds
- providing resources for students at all levels of ability including exceptional children
 - maintaining a collection of professional materials on subjects of interest to computer science teachers
 - providing instruction in electronic search techniques for available resources such as the Internet and databases
 - providing interdisciplinary experiences to help students comprehend and anticipate the links between computer science and other disciplines and areas of study.

Gender Equity

Saskatchewan Education is committed to providing quality education for all students in the K to 12 system. It is recognized that expectations, if based primarily on gender, limit students' ability to develop to their fullest potential. While some stereotypical views and practices have disappeared, others remain. Where schools have endeavoured to provide equal opportunity, continued efforts are required so that equality of benefit or outcome may be achieved. It is the responsibility of schools to decrease sex-role expectations and attitudes in an effort to create an educational environment free of gender bias. This can be facilitated by increased understanding and use of gender-balanced material and strategies, and further efforts to analyze current practice. Both girls and boys need encouragement to explore non-traditional as well as traditional options.

In order to meet the goal of gender equity in the K to 12 system, Saskatchewan Education is committed to bringing about the reduction of gender bias that restricts the participation and choices of students. It is important that the Saskatchewan curriculum reflects the variety of roles and the wide range of behaviours and attitudes available to all members of our society. This curriculum guideline strives to provide gender-balanced content and activities and to promote teaching strategies described in inclusionary language. These actions will assist teachers to create an environment free of bias and enable all students to share in experiences and opportunities that develop their abilities and talents to the fullest.

Teachers need to believe that both females and males can perform well in computer science. Teachers should also become aware of the attitudes displayed by their students and help them to view themselves as being able to achieve

in computer science. It is important to show students the relevance of computers in their lives, choosing examples that come from the experiences of all students, both male and female.

From an early age, students need to be made aware that daily living and many careers require an ability to master and work with modern technology. Teachers need to be sensitive to their interactions with students and ensure that everyone takes an active part in classroom activities. Being aware of interactions between students that may reinforce limiting behaviour or attitudes and taking opportunities to discuss them will help students to acquire a broader understanding of their own abilities and their potential. All of these actions will support and reinforce the principle of gender equity in a computer science context and move toward improved teaching practice.

Aboriginal Curriculum Content and Perspectives

The integration of Aboriginal content and perspectives into the K-12 curriculum fulfils a central recommendation of *Directions, Indian and Métis Education Action Plan* (1995) and *The Indian and Métis Education Policy from Kindergarten to Grade 12* (1995). In general, the policy states:

Saskatchewan Education recognizes that the Indian and Métis peoples of the province are historically unique peoples and occupy a unique and rightful place in society today. Saskatchewan Education recognizes that education programs must meet the needs of Indian and Métis peoples, and that changes to existing programs are also necessary to benefit all students.

In a pluralistic society, the inclusion of Indian and Métis perspectives benefits all students. Cultural representation in all aspects of the school environment helps to provide students with a positive group identity. Appropriate resources foster meaningful experiences for Indian and Métis students and promote the development of positive attitudes in all students towards Indian and Métis peoples. Awareness of one's own culture and the cultures of others develops positive self-concept and enhances learning.

Saskatchewan Indian and Métis students come from varied cultural backgrounds and geographic areas encompassing northern, rural, and urban

environments. Teachers must be given support that enables them to create instructional plans relevant to meeting diverse needs. Varied social, cultural, and linguistic backgrounds of Indian and Métis students imply a range of strengths and learning opportunities for teachers to develop. Explicit guidance, however, is needed to assist teachers in meeting the challenge by enabling them to make appropriate choices in broad areas of curriculum support. Anti-bias curricula, cross-cultural education, first and second language acquisition, and standard and non-standard usage of language are becoming increasingly important to classroom instruction. Care must be taken to ensure teachers utilize a variety of teaching methods that build upon the knowledge, cultures, and learning styles students possess. All curricula, including computer science, require adaptations to the content, instructional practices, and learning environment that reflect the needs of the students.

The following four points summarize the Department's expectations for the appropriate inclusion of Indian and Métis content in curriculum and instruction.

- Curricula and materials will concentrate on positive images of Indian, Métis, and Inuit peoples.
- Curricula and materials will reinforce and complement the beliefs and values of Indian, Métis, and Inuit peoples.
- Curricula and materials will include historical and contemporary issues.
- Curricula and materials will reflect the legal, political, social, economic, and regional diversity of Indian, Métis, and Inuit peoples.

The Computer Science Curriculum Guidelines support the expectations of Indian and Métis content and perspectives by:

- encouraging teachers to present programming projects related to the students' environment
- utilizing activity-based, hands-on approaches that promote success and the development of a positive self-image
- utilizing materials, both concrete and pictorial, with which the student is familiar and comfortable
- incorporating student-focused projects
- demonstrating the applicability of computer science through integration with other areas of study and daily life
- suggesting students work together in cooperative groups of varying sizes.

The final responsibility for accurate and appropriate inclusion of Indian and Métis content in instruction rests with teachers. They have the responsibility of evaluating resources for bias and teaching students to recognize bias. The Computer Science Curriculum Guidelines provide teachers with opportunities to begin these integration and evaluation processes. The document *Diverse Voices: Selecting Equitable Resources for Indian and Métis Education* (Saskatchewan Education, 1995) provides support for teachers in evaluating resources for bias.

Instructional Approaches

It is necessary for teachers to use a broad range of instructional approaches to give students a chance to develop their understandings and abilities to investigate, to make sense of, and to construct meanings from new situations; to make and provide arguments for conjectures; and to use a flexible set of strategies to solve problems. In addition, greater opportunities can be provided for small-group work, independent learning, electronic networking, peer instruction, and whole-class discussions in which the teacher serves as a moderator.

Such instructional methods will require the teacher's role to shift from dispensing information to facilitating learning. New topics, whenever possible, should be introduced through real-life problem situations that encourage students to explore, formulate and test conjectures, prove generalizations, and discuss and apply the results of their investigations. As a result of such instruction, students should be able to learn computer science both creatively and independently and thereby strengthen their confidence and skill in doing computer science.

In fact, problem solving should not only be a means of instruction but also a goal. The relationship of problem solving to other teaching strategies is very fundamental. One way students can obtain practice in using a problem-solving process is for the learning situation to be one where they can discover for themselves the computer science they are to learn. *Instructional Approaches: A Framework for Professional Practice* (1991) provides additional information to understand and implement a variety of approaches to teaching.

The use of technology in instruction facilitates the teaching and learning of computer science. Computer software can be used for class demonstrations and independently by students to

explore additional examples, to perform independent investigations, to generate and summarize data as part of a project, or to complete assignments. More information on using a variety of instructional strategies can be found in the Instructional Strategies Series of booklets published by SIDRU and SPDU.

Assessment and Evaluation

Why Consider Assessment and Evaluation?

Much research in education around the world is currently focusing on assessment and evaluation. It has become clear, as more and more research findings accumulate, that a broader range of attributes needs to be assessed and evaluated than has been considered in the past. Assessment and evaluation are best addressed from the viewpoint of selecting what appears most valid in meeting prescribed needs.

Evaluations may focus on progress in student learning (student evaluation), the effectiveness of school programs (program evaluation), and the effectiveness of the curriculum (curriculum evaluation). Teachers also reflect on the effectiveness of their instruction (teacher self-evaluation). In *Student Evaluation: A Teacher Handbook* (1991), the difference among the various forms of evaluation is explained.

Student Evaluation

Teachers are encouraged to evaluate students on a wide range of learnings, through the use of a number of assessment strategies or techniques. Observation, conferencing, oral and written assignments, portfolios of computer science work, and performance stations are examples of useful non-traditional tools for assessing students' computer science understandings, skills, and attitudes.

The evaluation of student progress has a strong influence on both teaching and learning. Because the two main purposes of assessment are to help in making instructional decisions and to monitor student progress, assessment must be integrated into instruction. It helps to determine whether the teacher's intended meaning is the same as the student's constructed meaning and if it is not, the teacher can change the instruction to bring them together. If used appropriately, evaluation can promote learning, build confidence, and develop students' understanding of themselves.

Program Evaluation

Program evaluation is a systematic process of gathering and analyzing information about some aspect of a school program in order to make a decision or to communicate to others involved in the decision-making process. Program evaluation can be conducted in two ways relatively informally at the classroom level, or more formally at the classroom, school, or school division levels.

To support formal school-based program evaluation activities, the Department has developed the *Saskatchewan School-Based Program Evaluation Resource Book* (1989) to be used in conjunction with an inservice package.

Curriculum Evaluation

During the 1990s, new curricula continue to be developed and implemented in Saskatchewan. Consequently, there is a need to know whether these new curricula are being effectively implemented and whether they are meeting the needs of students. Curriculum evaluation, at the provincial level, involves making judgments about the effectiveness of provincially authorized curricula.

It involves gathering information and making judgments or decisions based on the information collected, to determine how well the curriculum is performing. The principal reason for curriculum evaluation is to plan improvements to the curriculum. Such improvements might involve changes to the curriculum document and/or the provision of resources or inservice to teachers.

Curriculum evaluation is described in greater detail in *Curriculum Evaluation in Saskatchewan* (Revised 1994).

Teacher Self-Evaluation

There are two levels of teacher self-evaluation: reflection on day-to-day classroom instruction, and professional self-evaluation.

Teachers refine their skills through reflecting upon elements of their instruction which includes evaluation. The following questions may assist teachers in reflecting on their evaluations of student progress.

- Was there sufficient probing of student knowledge, understanding, skills, attitudes, and processes?
- Were the assessment strategies appropriate for the student information required?
- Were the assessment conditions conducive to the best possible student performance?
- Were the assessment strategies fair/appropriate for the levels of student abilities?
- Was the range of information collected from students sufficient to make interpretations and evaluate progress?
- Were the results of the evaluation meaningfully reported to students, parents, and other educators as appropriate?

Through reflection on questions like those above, teachers are able to improve their strategies for student evaluation.

It is also important for teachers, as professionals, to engage in self-evaluation. Teachers should take stock of their professional capabilities, set improvement targets, and participate in professional development activities. Some ways teachers can address their professional growth are by: reflecting on their own teaching; reading professional documents (e.g.: articles, journals, and books); attending workshops, professional conferences, and courses; and, developing networks with other professionals in their fields.

Clarification of Terms

To enhance understanding of the evaluation process, it is useful to distinguish between the terms "assessment" and "evaluation". These terms are often used interchangeably causing some confusion over their meaning. Assessment is a preliminary phase in the evaluation process. In this phase, various techniques are used to gather information about student progress. Evaluation is the weighing of assessment information against some standard (such as a curriculum learning objective) in order to make a judgment. This may then lead to decision and action.

There are three main types of student evaluation: diagnostic, formative, and summative.

Diagnostic evaluation is used to determine where instruction should begin and usually occurs at the beginning of the school year or before a unit of instruction. Its main purposes are to identify students who lack prerequisite knowledge, understanding, or skills, so that remedial help can be arranged; to identify gifted learners to ensure they are being sufficiently challenged; and, to identify student interests. It may consist of teacher-designed tests, informal discussions, and tasks that incorporate the pre-requisite skills or concepts. Diagnostic evaluation provides information essential to teachers in designing appropriate programs for students.

Formative evaluation is an ongoing classroom process that keeps students and educators informed of students' progress towards program learning objectives. It provides teachers with valuable information upon which instructional modifications can be made. Analysis of work samples, observation, interviews, checklists, and teacher-made tests are ways of collecting data for formative evaluation. This type of evaluation helps teachers understand the degree to which students are learning the course material and the extent to which their knowledge, attitudes, skills, and understandings are developing. Students are provided direction for future learning and are encouraged to take responsibility for their own progress.

Summative evaluation occurs most often at the end of the unit of study. Its primary purposes are to determine what has been learned over a period of time, to summarize student progress, and to report on progress to students, parents, and educators. It is a formal evaluation and students are informed in advance of the timing, method, and specific knowledge, skills, or behaviours being evaluated. Evaluation techniques include testing, analysis of work samples, interviews, checklists, and projects.

Seldom are evaluations strictly formative or summative. For example, summative evaluation can be used formatively to assist teachers in making decisions about changes to instructional strategies or other aspects of students' learning programs. Similarly, formative evaluation may be used to assist teachers in making summative judgments about student progress. However, it is important that teachers make clear to students the purpose of assessments and whether they will later be used summatively.

Typically, teachers conduct all three types of evaluation during the course of the school year.

Phases of the Evaluation Process

Evaluation can be viewed as a cyclical process including four phases: preparation, assessment, evaluation, and reflection. The evaluation process involves the teacher as a decision maker throughout the four phases.

In the **preparation phase**, decisions are made that identify what is to be evaluated, the type of evaluation (diagnostic, formative, or summative) to be used, the criteria against which student learning outcomes will be judged, and the most appropriate assessment techniques with which to gather information on student progress. Two main questions in this phase are "What do I want the students to know and be able to do?" and "What are valuable ways to ascertain whether they understand and can perform these activities?". The teacher's decisions in this phase form the basis for the remaining phases and may be made in consultation with the student.

During the **assessment phase**, the teacher identifies information-gathering strategies, collects student products, constructs or selects instruments, administers them to the student, and collects the information on student learning progress. The evidence gathered should link directly with the decisions made in the first phase. The teacher needs to ask questions like "What kinds of tasks can guide instruction and help monitor students' progress?" and "What are some important aspects to consider in choosing tasks?". The identification and elimination of bias (such as gender and cultural) from the assessment techniques and instruments, and the determination of where, when, and how assessments will be conducted are examples of important considerations for the teacher.

During the **evaluation phase**, the teacher interprets the assessment information and makes judgments about student progress. Teachers need to reflect on the assumptions they bring to the interpretation of evidence gathered. Based on the judgments or evaluations, teachers make decisions about student learning programs and report on progress to students, parents, and appropriate school personnel.

The **reflection phase** allows the teacher to consider the extent to which the previous phases in the evaluation process have been successful. Specifically, the teacher evaluates the utility and appropriateness of the assessment strategies used. Such reflection assists the teacher in making

decisions concerning improvements or modifications to subsequent teaching and evaluation.

All four phases of the evaluation process are included in diagnostic, formative, and summative evaluation.

Principles of Evaluation

Because evaluation is an integral part of the curriculum, Saskatchewan Education has developed the following guiding principles to provide a framework to assist teachers in planning for student evaluation.

- Evaluation is an essential part of the teaching-learning process. It should be a planned, continuous activity that is closely linked to both curriculum and instruction.
- Evaluation should be guided by the intended learning objectives/outcomes of the curriculum and a variety of assessment strategies should be used.
- Evaluation plans should be communicated in advance. Teachers must explain criteria and assessment techniques to students clearly and ahead of actual assessment. Students should have opportunities for input to the evaluation process.
- Evaluation should be fair and equitable. It should be sensitive to family, classroom, school, and community situations; it should be free of bias. Students should be given opportunities to demonstrate the extent of their knowledge, understanding, skills, and attitudes.
- Evaluation should help students. It should provide positive feedback and encourage students to participate actively in their own learning. Students may set standards of accomplishment and practise peer and self-assessment.

Student Assessment in Computer Science

Specific assessment strategies are selected or devised to gather information related to how well students are achieving the learning objectives of the curriculum. The assessment strategies used at any given time will depend on several factors such as the type of learning outcomes (knowledge, skill, attitude, understanding, value, or process),

the subject area content, the instructional strategies used, the student's level of development, and the specific purpose of the evaluation.

The foundational objectives and the learning objectives for the curriculum become the criteria for assessing students. Although these objectives should be attainable by the majority of students, that is not true for all. In addition, some students may not reach full potential because they are not challenged but are allowed to remain at the acceptable "average". The Adaptive Dimension recognizes that the needs of all students must be considered in order for effective teaching and learning to occur. Therefore, adaptations to instruction or procedures may be required. Evaluation must be made in a context that is significant and similar to the learning environment; e.g., applications of "conditionals and looping" concepts should be based on prior understanding. The teacher should be aware that some students with computer skills and abilities may have difficulty demonstrating these in a written assessment. In the choice of objectives and evaluation strategies, the teacher should consider the individual needs of students.

Learning computer science is a cumulative process that occurs as experiences contribute to understanding. A numerical grade offers only a glimpse of a student's knowledge. If the goal of assessment is to obtain a valid and reliable picture of a student's understanding and achievement, evidence must come from a variety of sources. These sources may include oral presentations, written work, observations, or various combinations of these. Examples of written work include programming projects, journals, essays or presentations, quizzes, and exams. Records of a student's progress may include anecdotal records, portfolios, and journals. Rating scales and observations checklists are also helpful devices to record evidence of a student's continued growth in understanding. The advantage of using several kinds of assessments is that a student's understanding can be continuously monitored. In addition, because students differ in their perceptions and thinking styles, it is crucial that they are given the opportunity to demonstrate their individual capabilities. A single type of assessment can frustrate students, diminish their self-confidence, and make them feel anxious about computer science.

The assessment of a student's computer science knowledge includes his/her ability to solve

problems, to use the language of computer science, to reason and analyze, to comprehend the key concepts and procedures, and to think and act in positive ways. Assessment should also examine the extent to which students have integrated and made sense of computer science concepts and procedures and whether they can apply these concepts and procedures to situations that require creative and critical thinking.

Understanding concepts and their interrelationships is essential to interpreting a situation and deriving an appropriate plan of action. Knowing what procedures are appropriate and how to execute them is essential to carrying out the plan successfully.

Progress in problem solving should be assessed systematically, deliberately, and continually. Methods for assessing a student's ability to solve problems include observing the student solving problems individually, in small groups, or in class discussions. Other methods include listening to students discuss their problem-solving processes and analyzing other computer programs. A rating scale is useful for assessing a student's problem-solving skills. It may include rating a student's willingness to engage in problem solving, the use of a variety of strategies, facility in finding the solution to problems, and consistency in verifying the solution.

Assessment of a student's ability to communicate includes the meaning she/he attaches to the concepts and procedures of computer science. It also involves his/her ability in talking about, writing about, understanding, and evaluating computer science ideas. In assessing a student's ability to communicate, attention should be given to the clarity, precision, and appropriateness of computer science terms and symbols. Teachers should ask questions regularly. Discussion is also an excellent means of judging a student's ability to function as a critical participant in small groups or in the class as a whole. Much of the assessment of communication skills can be done by oral and informal methods.

The assessment of a student's ability to use computers and software is paramount. A student's ability to structure and present information with the use of technology can be assessed by various means.

An understanding of computer science concepts involves more than mere recall of definitions and recognition of examples. It also encompasses a broad range of abilities. Assessment must include

the aspects of conceptual understanding by focusing on a student's ability to discriminate between relevant and irrelevant attributes of a concept by selecting examples and non-examples, to represent concepts in various ways, and to recognize their various meanings. Observational checklists, anecdotal records, or written reports may be used to assess such conceptual understanding.

Learning computer science also includes developing a positive attitude towards computer science. The assessment of a student's attitude requires information about her/his thinking and actions in a wide variety of situations. Encourage students to reflect on their own thinking. Students' attitudes are reflected in how they ask and answer questions, work on projects, and approach new tools and techniques. Observations, programming projects, journals, and oral presentations are all excellent ways to assess a student's attitude towards computer science.

The computer science curriculum incorporates a number of instructional strategies and methods. Each one suggested has been chosen to facilitate the learning objectives effectively. Teachers should choose the most appropriate assessment techniques for what they have taught and how they have taught it. Computer science involves more than just creating a program that works. The strategy, the procedure used to plan it, and the communicative skill employed to convey understanding are also important. Evaluation identifies for the students, as well as their parents, those aspects of learning that are valued. The curriculum guidelines identify the valued objectives, hence learnings.

Some assessment techniques are better suited for providing certain kinds of information than others. The techniques the teacher uses will depend upon the purpose of the assessment. For example, if a teacher wishes to assess the knowledge the students have gained, a quiz or test using objective items would provide the necessary evidence of student learning. However, should the teacher wish to assess problem-solving skills, a performance test with specific criteria listed on a rating scale or checklist as a recording instrument may prove of more value. Should information on student attitude or work involvement be the purpose of the assessment, anecdotal records may be the best device. With processes that students would use in performing tasks, a teacher may choose to observe students during performance assessments or presentations

and record the information on checklists or rating scales listing specific areas to be assessed. Teachers do not necessarily need to assess all students every class. For example, a teacher may prepare anecdotal records for five students per class.

Examples of how instructional strategies and assessment techniques may be applied to this curriculum can be found throughout these guidelines in the Notes column. These suggestions are not meant to be prescriptive. They are offered as examples of how a teacher may use a range of assessment techniques to assess student progress in a variety of different areas. There are many examples in Information Processing and other Core Curricula that may be used or adapted for student assessment. For further information on the various assessment strategies and types of instruments that can be used to collect and record information about student learning, refer to *Student Evaluation: A Teacher Handbook* (1991).

Record Keeping

An important aspect of organizing an evaluation plan is managing the records that are kept.

The following ideas for record keeping and organizing for assessment have been adapted from *Student Evaluation: A Teacher Handbook Follow-up Inservice* (1993). These tips may assist any teacher when considering how an evaluation program may be organized.

Keep a master list of the skills, attitudes, and processes that may be applicable to all subject areas. This will save time in constructing assessment instruments.

Involve students in assessing their own learning progress. When students are aware of the expectations the teacher has for them, the students are able to become accurate self-assessors.

Have students file assessment instruments (as much as they are able). Students will learn organizational skills as well as learn to become fully aware of the information being gathered on their progress.

Extend observations over time. It is not possible nor advisable to see every observable student behaviour that may be listed on a checklist or rating scale exhibited by every student during one period of time. A checklist completed over a

term, used on a regular basis, gives a better view of what a student can do compared with observations that are one-time, one-shot attempts.

Keep it simple. Consider the essentials and streamline the assessment routine.

Assigning Grades

Traditional grading systems often use a narrow focus on whether an answer is right or wrong. With holistic scoring, grades can be based on a student's total ability, content knowledge, and processes rather than on the percentage of correct answers. Criteria can be determined for a rubric that can be used to score the work but can also communicate to students what the score means and how they can improve their performance. A score that indicates to the students a grading of their understanding, solution(s), and explanation(s) is much more valuable than a single score. During the course of the year, these scores can be monitored and a clear picture of the student's progress in specific areas of thinking can be created.

The grades given should be fair, objective, and an accurate reflection of the student's achievement. Grading is a judgmental, and therefore, subjective process, hence teachers must be cautious not to evaluate only a small part of the student's knowledge, skills, and abilities.

Sample Evaluation Scheme

Program Assignments	40%
Examinations and quizzes	30%
Research Project	10%
Final Examination	20%

Informing Students and Parents or Guardians about Evaluation

"Best evaluation practice" stipulates that students know at the outset what will be assessed, how it will be assessed, why it is to be assessed, when it will be assessed, and how the assessment will contribute to an evaluation of their progress.

Students should know criteria, performance levels, and what constitutes high quality work. This enables them to evaluate their own work and know what the expectations are.

Whenever possible, this information should also be communicated to parents or guardians.

Conclusion

Evaluation is the reflective link between what ought to be and what is, and therefore, it is an

essential part of the educational process.

Evaluation procedures should provide positive and supportive feedback to students and should assist in making good decisions about the next steps in their learning and instruction. Regard students as active participants and encourage the development of self-appraisal. The main purposes for evaluating are to facilitate student learning and to improve instruction. By continuously evaluating student progress, school programs, curriculum, and the effectiveness of instruction and evaluation, these purposes will be realized.

Evaluation instruments should mirror the class experience as much as possible. While testing student progress using computers can be problematic, using only paper and pencil instruments is likely unfair.

Program Considerations

Programming Languages

Learning to program a computer is one of the important topics in the study of computer science, but it must be noted that there are many other important components of the science. The theoretical science of computation has its foundations in mathematics and linguistics, where algorithms for efficient computation and languages for expressing these algorithms are studied. The practice of computer programming grew from the capabilities of electronic computers to physically execute algorithms and interpret languages.

The main activity of most computer scientists is to design, develop, and program software systems. This activity has recently become known as "software engineering". Improving methods for designing and developing software that is correct, reliable, safe, and efficient has been a major activity in computer science over the past 50 years. Software design methodologies have progressed rapidly during the past 15 years. New paradigms such as object-oriented programming and rapid application development environments have changed the way software is designed and developed.

Computer Science 20 and 30 are more than computer programming courses. They are based on a broader range of objectives. In this approach to computer science, a programming language is used as a tool. However, some programming language must be chosen over others. It is desirable for students to have experience with more than one

language so that they are aware that commonalities exist among different languages.

A different language should be used in the two classes. The teacher should select the languages so that the one used at the 30 level presents a different perspective so that students get as broad a range of experience as possible. It is recommended that an object-oriented language be used in at least one of the courses.

The selection of programming languages is left to the teacher. However, situations may arise where guidance is needed. For those cases, **suggested languages are listed in Appendix A**. These are recommended for beginners because they are good languages that are easy to obtain, cheap, and relatively easy to use.

The concepts learned at the 20 level will be covered readily with the second language at the 30 level. It is known from research that students transfer their knowledge of one computer language to the learning of another. This transfer occurs, in part, by analogy. Features of the new language (syntax) are learned better when the teacher points out their similarities with features of the previous language. Analogies are also important when presenting control and data structures of the new language. Students will learn the new structures better when they are shown the similarities and differences with the previous language. Students will continue to explain structures of the new language in terms of the previous language, even after they become familiar with the new. Teachers can utilize this link with the previous language as an instructional aid for some time after the new language is introduced.

By the time students complete Computer Science 30, they should be exposed to a variety of computer experiences. Students should gain an understanding of the software methodology compatible with the new language. All programming projects should promote good software design habits, using a recognized software design methodology and producing a well-documented and tested software system.

Problem Solving

Various Core topics have been identified as important for Computer Science 20 and 30. Suggested times for these topics are indicated reflecting the level of emphasis. Optional topics may be chosen depending on interest and time available.

Integration

The Internet and multimedia presentations may be used by students in other classes. This would provide opportunities for integration of subjects. The Computer Science class could act as a resource for understanding the uses in other classes.

Assignments

Assignments, intended to be completed in class or at home, enhance students' understanding, skills, and proficiency in computer science. Assignments can ensure that each Secondary Level credit accommodates 100 instructional hours.

The research component of the courses gives students an opportunity to explore the current state of knowledge in a Computer Science area of interest. It will also give them opportunity to gain experience with state of the art presentation techniques.

Care must be taken to assure that assignments are meaningful extensions of the concepts taught in class. Unimaginative assignments can often inhibit students' creativity, love of computer science, and desire to extend their learning independently. Assignments should develop students' higher levels of thinking by being structured in a problem-solving mode so that students have the opportunity to apply the computer science concepts learned. Although the objectives are presented in a numbered sequence, it is not required that they be presented in this order. The teacher should present the objectives in a way that is appropriate for the situation in the class.

Parents/caregivers can be significant contributors to this learning process. Opportunities for parents/caregivers to be involved in the data collection and problem-solving processes allow them to display interest in the student's work. It also offers them the opportunity to become familiar with the student's program.

References

- Alberta Education. (1994 Interim). *Career and technology studies: Computer processing 10-20-30*. Edmonton, AB: Author.
- Saskatchewan Education. (1995). *Connections to a world of learning. Saskatchewan's multimedia learning strategy*. Regina, SK: Author.
- Saskatchewan Education. (1995). *Indian and Métis education policy from kindergarten to grade 12*. Regina, SK: Author.
- Saskatchewan Education. (1993). *Student evaluation: A teacher handbook follow-up inservice*. Regina, SK: Author.
- Saskatchewan Education. (1992). *The adaptive dimension in core curriculum*. Regina, SK: Author.
- Saskatchewan Education. (1992). *Gender equity: A framework for practice*. Regina, SK: Author.
- Saskatchewan Education. (1992). *Saskatchewan school-based program evaluation resource book*. Regina, SK: Author.
- Saskatchewan Education. (1991). *Instructional approaches: A framework for professional practice*. Regina, SK: Author.
- Saskatchewan Education. (1991). *Student evaluation: A teacher handbook*. Regina, SK: Author.
- Saskatchewan Education. (1988). *Understanding the common essential learnings: A handbook for teachers*. Regina, SK: Author.
- Saskatchewan Education. (1987). *Resource-based learning. Policy, guidelines, and responsibilities for Saskatchewan learning resource centres*. Regina, SK: Author.
- Saskatchewan Education. (1984). *Computer applications 10, 20 and computer science 10, 20, 30. A curriculum guide for division IV*. Regina, SK: Author.
- Saskatchewan Education. (1984). *Directions: The final report*. Regina, SK: Author.

Computer Science 20

Outline of Foundational and Learning Objectives

Unit 1: Software and Hardware

Foundational Objective

To familiarize the students with the software and hardware components that comprise a computer system and emphasize safe, responsible use of them.

Suggested Time: 5-10 hours

Learning Objectives cover:

- 1.1 Responsible Computer Use
- 1.2 Operating System Use
- 1.3 Operating System Analysis
- 1.4 Hardware Use and Maintenance

Unit 2: Problem Solving

Foundational Objective

To provide the students with an understanding of problem-solving methods and techniques.

Suggested Time: 5-10 hours

Learning Objectives cover:

- 2.1 Problem Analysis
- 2.2 Problem Solving: Strategies
- 2.3 Algorithms: Describing Output
- 2.4 Algorithms: Correcting Errors

Unit 3: Fundamentals of Programming and Design

Foundational Objective

To be familiar with and understand the fundamentals of computer program writing and software design.

Suggested Time: 5-10 hours

Learning Objectives cover:

- 3.1 Program Structures
- 3.2 Benefits of Structured Programming
- 3.3 Modular Program Structure
- 3.4 Program Testing/Debugging
- 3.5 Internal Program Documentation
- 3.6 External Program Documentation

Unit 4: Experience with Programming and Design

Foundational Objective

To provide hands-on experience with program writing and software design and interpreting and writing documentation.

Suggested Time: 50-65 hours

Learning Objectives cover:

- 4.1 Initial Concepts
- 4.2 Output Statements
- 4.3 Variables and Assignments
- 4.4 Assign and Output
- 4.5 Input Data to a Variable
- 4.6 Anatomy of a Program
- 4.7 Concepts: Conditionals and Looping
- 4.8 Programming: Conditionals and Looping
- 4.9 Programming: Procedures and Subprograms
- 4.10 Programming: Single and Nested Loops
- 4.11 String Functions
- 4.12 Numeric Functions
- 4.13 User-Defined Functions
- 4.14 Concepts: Single-Dimension Arrays

4.15 Programming: Single-Dimension Arrays

Unit 5: Networks (Optional)

Foundational Objective

To familiarize the students with the software and hardware systems that comprise the different types of networks and emphasize safe, responsible use of them.

Suggested Time: 5 hours

Learning Objectives cover:

5.1 Networks

5.2 Understanding the Internet

5.3 Using The Internet

5.4 Multimedia

Unit 6: Careers Related to Computer Science

Foundational Objective

To identify and describe computing careers, and the necessary preparations for them.

Suggested Time: 5 hours

Learning Objective covers:

6.1 Investigating Careers in Computer Science

Unit 7: Research Topics

Foundational Objective

To provide hands-on experience with current research and presentation methods that utilize information technologies through investigation of a computer science topic of interest.

Suggested Time: 7-10 hours

Learning Objective covers:

7.1 Researching a computer science topic and presenting the findings to peers using computer technology.

Unit 1: Software and Hardware

Suggested Time: 5-10 hours

Foundational Objective

- To familiarize the students with the software and hardware components that comprise a computer system and emphasize safe, responsible use of them.

Common Essential Learnings Foundational Objective

- To understand and use the computer as a helpful learning tool. (TL)

Note: other CELs may be emphasized here.

Learning Objectives	Notes
1.1 Demonstrate acceptable attitudes and responsible practices by: <ul style="list-style-type: none">• observing copyrights of software publishers• using school computers and networks in a responsible manner, both in the physical care of the computer and the nature of the uses of it• using computers in an appropriate way. (PSVS)	Schools must model proper attitudes and practices in use of software. All software in use in a school should be properly licensed. A police officer, a software vendor, a software author or a lawyer would be a good resource person to talk about software piracy. It would be beneficial to have students discuss the division or school Technology Use Policy. If one does not exist, students could create a draft version.
1.2 Be able to demonstrate understanding of the fundamentals of the operating system that is on the computer being used. Students will be able to perform the following operations: <ul style="list-style-type: none">• start and shut down the system• write, copy, erase, edit and print files• run applications, specifically the programming language• interpret the system software manuals and copyright agreement.	If an extra computer is available, have students install an operating system on it as an individual or team assignment. The system could be removed in preparation for the next group.
1.3 Contrast the characteristics of the operating system being used with one other. (TL)	If only one operating system is in use in the school, a community resource person, teacher or student who uses a different system could provide some information about another system.

Learning Objectives

Notes

1.4 Demonstrate an understanding of the basic workings of microcomputer hardware. Students will be able to do the following things:

- describe the proper procedures and precautions to work safely inside the case of a computer
- describe the input and output devices of the system
- properly disconnect and ground a computer, remove the cover, locate and describe the function of
 - CPU chip
 - power supply
 - storage devices
 - memory chips
 - expansion slots and boards
- remove a board from a computer and re-install it.
(TL)

A kit from Intel, **The Journey Inside: The Computer** is available free of charge to teachers. It was designed for Middle Level students and may be too elementary for senior students. Contact <http://www.intel.com/intel/educate/teacher/journey/index.htm>.

This activity should be done **only** if hardware availability allows it. An older computer such as those available from the **Computers for Schools** program could be set aside for this activity. Safety of students and proper grounding of components should be stressed; static electricity can be deadly to computer chips. Students should be cautioned that doing this on other computers might be dangerous.

Unit 2: Problem Solving

Suggested Time: 5-10 hours

Foundational Objective

- To provide the students with an understanding of problem-solving methods and techniques.

Common Essential Learnings Foundational Objective

- To understand how knowledge is constructed in the sciences. (CCT)

Note: other CELs may be emphasized here.

Learning Objectives	Notes
2.1 Analyze a programming task or solution in top-down form and explain the procedures for solving the problem in the form of an algorithm (flowchart or pseudo code).	These objectives may be presented before students begin programming. However, they may be integrated with other learning objectives in programming projects.
2.2 State and explain the steps of a problem-solving strategy for the solution of computing problems. (COM)	Concrete examples are useful to introduce the concept of algorithms: <ul style="list-style-type: none">• create a sandwich• change a tire• change a light bulb A teacher could play the role of a Martian taking the instructions of the students extremely literally while performing the task.
2.3 To trace through an algorithm (flowchart or pseudocode) and describe correctly the specific output of a given algorithm for a given set of data and describe in general what it accomplishes. (CCT)	These objectives should be referred to throughout all other programming projects.
2.4 Identify and correct errors in a given algorithm.	

Unit 3: Fundamentals of Programming and Design

Suggested Time: 5-10 hours

Foundational Objective

- To be familiar with and understand the fundamentals of computer program writing and software design.

Common Essential Learnings Foundational Objective

- To reflect upon how knowledge is developed and changed in computer science. (CCT)
- To develop and use the vocabulary associated with computer programming. (COM)

Note: other CELs may be emphasized here.

Learning Objectives	Notes
3.1 Demonstrate an understanding of three basic programming structures: sequence (action block), repetition (loop block) and selection (branch block).	These principles may be introduced before students begin creating programs. The teacher might prefer to incorporate them with the Learning Objectives of Unit 4. Students should be reminded of these objectives throughout their programming projects.
3.2 Demonstrate an understanding and appreciation of the benefits of a structured programming style. (CCT)	The instructor should insist that these fundamentals of good programming be present in all student projects. The instructor should be sure to practice, as well as preach, this style.
3.3 Demonstrate understanding of programs composed of a main logic module and sub-modules linked hierarchically to it (each module having one point of entry and one exit).	It is a good idea to have students transfer algorithm or pseudocode in a new program first as comments or remarks. The students can then fill in the necessary code under the appropriate comment. This will promote a proper structured approach to a problem and prevent the common practice of putting comments in a program after it is finished.
3.4 Demonstrate the ability to test and debug a program by: <ul style="list-style-type: none">• using manual testing ("program tracing") techniques with paper and pencil• placing position markers within the program for the output of intermediate results. (CCT)	Students should understand the difference between a logical error and a syntax error.

Learning Objectives

Notes

- 3.5 Demonstrate an understanding of appropriate internal documentation and the use and value of:
- imbedded comments
 - descriptive variable names
 - indentation
 - spacing for clarity (on-going)
 - defining global and local variables.
- 3.6 Demonstrate an understanding of appropriate external documentation of the use and value of:
- showing the interrelationships of modules through the use of a diagram delineating the input/output formats
 - providing a program description
 - describing the limitations of the program. (COM)
- The instructor should consider these things when assessing and evaluating student program code.
- All students should write a Users' Manual for at least one program. Have the program and manual tested by someone unfamiliar with the program. This could be an interesting opportunity for peer review and assessment.

Unit 4: Experience with Programming and Design

Suggested Time: 50-65 hours

Foundational Objective

- To provide hands-on experience with program writing, software design, and interpreting and writing documentation.

Common Essential Learnings Foundational Objectives

- To use formal procedures to learn about programming language developments. (TL)
- To apply mathematical concepts during computer science language exercises. (NUM)

Note: other CELs may be emphasized here.

	Learning Objectives	Notes
4.1	Demonstrate general knowledge of programs in the language, including: <ul style="list-style-type: none">• use of internal documentation statements• order of execution of statements• executable and non-executable statements• termination of a program. (TL)	The teacher should explain the value of these things in a program. It can be valuable for students to critique examples of good and bad program code.
4.2	Write programs in the language that outputs: <ul style="list-style-type: none">• blank lines• numbers• strings (messages)• items that are formatted and spaced in a variety of ways• strings and numbers on one line. (TL, COM)	It is a good idea to have some sort of “ice breaker” activity to reduce the anxiety level for beginning programmers. Writing the first program can be daunting. Students should be given an explanation of the criteria which will be used when their programs are evaluated. Creating a checklist of these for the students to use is good. Whenever possible, have student programming projects link to other courses such as science, mathematics or accounting. The instructor can provide the students with test data to use with their program to see if the results are valid.

Learning Objectives

Notes

4.3 Demonstrate understanding and skill in the use of variables and assignment of values to variables, as follows:

- differentiate between variables and values
- identify primitive types of variables
- identify the characteristics of the different types of variables and the operators appropriate to the different types
- explain conventions for naming variables
- identify an acceptable program statement that assigns a value to a variable
- explain how a variable assignment is executed.

A concrete model is useful to demonstrate how a value is assigned to a variable.

- A **“RAM mailbox”** structure with several compartments is an excellent model. Objects representing information can be moved among the compartments to show how values are assigned and moved. References to the mailbox are useful for more advanced topics such as traces, arrays, and sorts.
- Role playing can be a good experience with students representing memory locations and data.

4.4 Write programs in the language that assigns an output, as follows:

- assign numbers to numeric variables
- assign strings to string variables
- transfer the value of one storage variable to another
- output the values of numeric and string variables in a formatted way. (NUM)

It is important that students understand the difference between numeric and string variables and the way they are handled in the language. An activity early in the course to help students learn the differences can prevent much grief for them. An assignment where students correct variable mismatches or write a program with mismatches for a classmate to correct might be valuable.

4.5 Write programs that:

- display appropriate messages when prompting the user
- store numeric data in an appropriate variable
- store string data in an appropriate variable.

Learning Objectives

Notes

- 4.6 Recognize the various structural components of a program including variable and type declarations, conditionals, loops, modules, procedures/sub-routines, functions and other components that the chosen language supports.
- 4.7 Demonstrate understanding and skill in the use of conditionals and looping, as follows:
- define a conditional, a loop
 - understand how infinite loops are created and why they should be avoided
 - describe the relational operators
 - use relational operators to compare numeric and string values.
- 4.8 Demonstrate an understanding in the use of the language's conditional and loop commands, as follows:
- state the purpose of each statement of the command(s)
 - identify and correct invalid uses of the command(s)
 - follow and diagram the working of a conditional (if statement) and series of nested conditionals identifying the operators and the logical concepts involved
 - follow and diagram the working of a loop, tracking counter values, etc.
 - explain nested loops and how they work.
- 4.9 Demonstrate understanding of the value and importance of proper use of sub-programs and of using sub-routines whenever practical within programming projects.

It can be a good idea to have the students write procedures for a program body supplied by the teacher.

Conditional statements in most programming languages are implemented with an **if-then-else** structure. In some older languages, **go to** statements were used to simulate this structure. However, the use of **go to** should be STRONGLY discouraged in all student programming. If your chosen language requires use of **go to** statements, it is time for an upgrade.

The instructor may decide to restrict the topic to one kind of loop structure; however, students should be aware that most languages have several loops, each of which has different attributes. For example, some loops are count based and some are condition based.

Nested loops can be demonstrated through role-playing. A student could perform an action loop, such as clapping five times. This action loop can be nested inside another such as standing up and sitting down.

The students may find an ongoing program, that is added to as new concepts are introduced, to be more interesting than a new program for every concept.

Learning Objectives	Notes
4.10 Write programs involving single and nested loops.	
4.11 Write programs that contain manipulations with strings using the string manipulation functions in the language. (COM)	String comparison, joining, case change, and dissecting should be discussed and practised. A model of a string variable can show that different string functions do not change the value of the variable, they just look at a piece of it.
4.12 Write programs that contain uses of the built-in numeric functions of the language.	
4.13 Demonstrate an understanding of and skill in the use of user-defined functions as follows: <ul style="list-style-type: none"> • identify appropriate uses • contrast characteristics with other types of routines and procedures. (NUM) 	It is useful to show the value of functions by contrasting two programs that produce similar output while differing by using or not using functions.
4.14 Demonstrate an understanding of and skill in the use of single dimension arrays as follows: <ul style="list-style-type: none"> • describe the way that subscripted variables are stored in memory • explain how a loop can deal with numerous variables using an array • explain how a variable can be used to select a particular element from an array. 	Visual models are valuable when introducing arrays to students. When beginning to use arrays, students will understand the processes much better if they first fill and extract data from an array with separate actions rather than doing it recursively. They will soon want a way to improve the process and will be eager to use loops to manipulate data in an array.
4.15 Use single-dimension arrays appropriately in a program, as follows: <ul style="list-style-type: none"> • enter data into an array • access data from an array • search and sort items in an array. 	The sort and search should be done quickly, with the purpose of demonstrating the use of arrays. It is not intended to have students become proficient with all aspects of searches here.

Unit 5: Networks (Optional)

Suggested Time: 5 hours

Foundational Objective

- To familiarize the students with the software and hardware systems that comprise the different types of networks and emphasize safe, responsible use of them.

Common Essential Learnings Foundational Objective

- To respond sensitively to the ideas, comments and products of others. (PSVS)

Note: other CELs may be emphasized here.

	Learning Objectives	Notes
5.1	Demonstrate an understanding of the nature of networks as follows: <ul style="list-style-type: none">• describe the components and workings of a local area network• explain the role of network manager• discuss network security issues• describe how local area networks can be connected into a wide area network. (TL)	If the school has a network, the network manager can be an excellent resource.
5.2	To demonstrate an understanding of the Internet as follows: <ul style="list-style-type: none">• be familiar with the nature and history of the Internet• explain how a connection is made to the Internet and the different protocols that make it work• demonstrate an understanding of the different functions of the Internet• understand and observe “netiquette”, the code of ethics of the Internet• discuss Internet security issues and viruses. (PSVS)	

Learning Objectives

Notes

- 5.3 Demonstrate capability to use the Internet by:
- connecting
 - sending and receiving e-mail
 - browsing the World Wide Web
 - using World Wide Web search tool
 - downloading a file by **file transfer protocol (ftp)** from an ftp site and examining it with appropriate software to ensure that it is safe and virus free. (TL, COM)
- Students may have covered this in other courses, in which case it may be omitted or covered in greater depth.
- Most computer languages have websites and/or newsgroups where users congregate. These could be a valuable resource for computer science students.
- This document is online as part of the **Evergreen Curriculum** at <http://www.sasked.gov.sk.ca/docs/evergrn.html>
- 5.4 Demonstrate an ability to do simple multimedia presentations by:
- creating a simple, linear multimedia presentation containing graphics
 - creating a simple page in World Wide Web format using **hypertext markup language (html)**. (TL, COM)
- Students may have covered this in other courses, in which case it may be omitted or covered in greater depth.
- This might be a good opportunity for subject integration; students could do a presentation for another class.

Unit 6: Careers Related to Computer Science

Suggested Time: 5 hours

Foundational Objective

- To identify and describe computing careers and the necessary preparations for them.

Common Essential Learnings Foundational Objective

- To use questions as tools to further their own and others' understanding about careers related to computer science. (COM)

Note: other CELs should be emphasized here.

Learning Objectives

Notes

6.1 Become aware of the potential jobs for people with training in computer science by investigating job descriptions, primary responsibilities associated with each, salaries, working conditions and educational requirements. (PSVS, IL)

Guest speakers and videos are good resources for this objective.

Post-secondary institutions often like to make contact with high school students. Many institutions have a presence on the World Wide Web.

Unit 7: Research Topics

Suggested Time: 7-10 hours (This will vary with class size and availability of resources.)

Foundational Objective

- To provide hands-on experience with current research and presentation methods that utilize information technology, while investigating a computer science topic of interest.

Common Essential Learnings Foundational Objective

- To seek information through a steadily expanding network of options, including other libraries, databases, individuals, and agencies. (IL)

Note: all CELs should be emphasized here.

Learning Objectives	Notes
7.1 To research a computer science topic and present the findings to peers using computer technology. (IL, COM)	<p>Work with students to create a list of suggested topics for the student research. Teachers may choose other topics; ones specific to their local community would be fine. The project should make up a substantial part of the coursework, with about 10% of the grade derived from it.</p> <p>Assignments should be made early in the course with the expectation that the students will present their findings to the class. It is important that the students use computer technology of some sort in the presentation of their results. Students should also be encouraged to use technological resources such as the Internet in their research.</p> <p>Some possible methods for students to deliver the information include:</p> <ul style="list-style-type: none">• seminar with technological aides• multimedia presentation• a series of web pages• tutorial program• model

List of Topics

Coaching of Novice Users

Prepare instructions and provide these in writing or orally.
Display sensitivity to the age and background of the novice.
"Adopt" a novice and accept the coaching duties.
Prepare a "Computer Coaching Do's and Don'ts" Report
(COM)

Creative Problem Solving

Describe the creative problem solving process.
Apply the process to real-world problems; work with a group.
Incorporate computers into some of the solutions.
(CCT, IL)

Documentation

Examine and criticize commercial documentation:

- tutorials
- user's guides
- programmer's manuals
- systems manuals

Field experience:

- research the characteristics of a user
- write documentation tailored to a particular user

(CCT, COM)

Fine Arts and Computers

Describe computer peripherals useful for artistic expression:

- experiment with a digitizer, light pen, touch tablet, mouse
- survey and describe graphical and "painting" software
- experiment with a synthesizer, MIDI interface

Produce art or music using a computer

Gaming and Computers

Describe the various functions of computers in gaming:

- opponents, partners, managers

Analyze the history of computer gaming

Relate past events to current trends

Survey current computer games:

- categorize
- analyze
- write a critique or review

Write computer game programs

Hackers and Security

Search the Web and create a report

History of Computing

Choose an example.

- the work of Charles Babbage and Ada Lovelace
- the work of Konrad Zuse
- the work of Turing
- development of ENIAC
- invention of the transistor
- invention of the integrated circuit (chip)
- growth of the Internet

Libraries and Computers

Report on uses of computers for library functions:

- cataloguing, circulation

Analyze the school library, and describe how computers might be involved:

- conduct a system analysis
- design a computerized library system

Project

Apply the programming skills gained in the course to a major project

- work on the components alone
- work with others, dividing the labour

Robotics

Outline the history of robotics

Describe how robots work

Discuss industrial applications of robots

Analyze the socio-economic implications of robots

Describe the effects of combining robots with artificially intelligent computers

Analyze current trends and predict future scenarios.

(TL)

School Applications of Computers

Interview elementary-level teachers:

- report on applications of computers to elementary education

Describe the role computers might play in school activities:

- sports, lunchroom, office, student council, library

User Interfaces

Examine and contrast the way in which different computer applications present information to users.

Compare programs with similar uses, (word processing) and also ones with different uses, (a spreadsheet and a graphics program).

Computer Science 30

Outline of Foundational and Learning Objectives

Unit 1: Software and Hardware Advanced

Foundational Objective

To extend the knowledge of software and hardware systems developed in Computer Science 20.

Suggested Time: 2-3 hours

Learning Objectives:

- 1.1 Software Use and Maintenance
- 1.2 Hardware Use and Maintenance

Unit 2: Problem Solving - Advanced

Foundational Objective

To provide the students with an opportunity to apply problem-solving methods and techniques.

Suggested Time: 3-4 hours

Learning Objectives:

- 2.1 General Problem Solving Methodology
- 2.2 Understanding the Problem
- 2.3 Conceiving a Model of the Problem
- 2.4 Building a Model of the Problem
- 2.5 Solving the Problem
- 2.6 Looking Back at the Solution

Unit 3: Problem Solving and Programming

Foundational Objective

To be familiar with and understand the methodology of problem solving in computer programs.

Suggested Time: 3-4 hours

Learning Objectives:

- 3.1 Describe a Problem Appropriately
- 3.2 Model a Problem Appropriately
- 3.3 Determine Problem Solution
- 3.4 Re-evaluate Problem Solution
- 3.5 Proper Structure of a Computer Program
- 3.6 Variables
- 3.7 Modules
- 3.8 Efficient Development of Computer Programs
- 3.9 High-level and Low-level Languages
- 3.10 Interpreted and Compiled Languages
- 3.11 Object Oriented Programming

Unit 4: Experience with Programming and Design - Advanced

Foundational Objective

To provide hands-on experience with program writing, software design and documentation writing by writing computer programs that will do the processes described.

Suggested Time: 50-65 hours

Learning Objectives:

- 4.1 Fundamental Concepts of the Language
- 4.2 Variables, Operators and Assignments
- 4.3 Expressions and Functions
- 4.4 Documentation of Programs
- 4.5 Input and Output Processes
- 4.6 Anatomy of a program
- 4.7 Conditionals and Looping
- 4.8 Arrays
- 4.9 Files

Unit 5: Number Systems and Codes (Optional)

Foundational Objective

To provide an opportunity for students to learn about the number systems and codes that are fundamental to computer processing of information.

Suggested Time: 5-8 hours

Learning Objectives:

- 5.1 Understanding ASCII Code
- 5.2 Using ASCII Code
- 5.3 Understanding the Binary Number System
- 5.4 Understanding the Hexadecimal Number System
- 5.5 Programming with Binary and Hexadecimal Systems

Unit 6: Impact of Information Technology

Foundational Objective

To provide students with an opportunity to explore the nature of computer science and the societal impact of information technology.

Suggested Time: 3-5 hours

Learning Objectives:

- 6.1 The Science of Computing
- 6.2 The Importance of Information
- 6.3 Economic Effects of Computers and Networks
- 6.4 Social Effects of Computers and Networks
- 6.5 Political Effects of Computers and Networks
- 6.6 Appropriate Role of Computers

Unit 7: Programming for Applications

Foundational Objective

To provide an opportunity for students to use their programming expertise in an application program.

Suggested Time: 3-4 hours

Learning Objective:

- 7.1 Programming within an Application

Unit 8: Internet and Multimedia (Optional)

Foundational Objective

To provide an opportunity for the students to increase their understanding of the Internet and to work with multimedia applications.

Suggested Time: 4-6 hours

Learning Objectives:

- 8.1 The Internet
- 8.2 Multimedia

Unit 9: Research Topics

Foundational Objective

To provide hands-on experience with current research and presentation methods that use information technologies through investigation of a computer science topic of interest.

Suggested Time: 7-10 hours

Learning Objective:

- 9.1 To research a computer science topic and present the findings to peers using computer technology.

Unit 1: Software and Hardware - Advanced

Suggested Time: 2-3 hours

Foundational Objective

- To extend the knowledge of software and hardware systems developed in Computer Science 20.

Common Essential Learnings Foundational Objective

- To understand and use the computer as a helpful learning tool. (TL)

Note: other CELs should be emphasized here.

Learning Objectives	Notes
<p>1.1 Extend knowledge of the software on the computer system being used in order to:</p> <ul style="list-style-type: none">• understand and discuss copyright issues with regard to software use and copying; understand the nature of software upgrades and the version of the software package students are using• understand the importance of backing up data• understand the nature and hazards of viruses and the importance of proper prevention and cure• be knowledgeable about the graphics, sound and multimedia capabilities of the system being used.	<p>Some parts of this objective may have been covered in other courses. Do an assessment first. A review may suffice.</p> <p>Schools must model proper attitudes and practices in use of software. All software in use in a school should be properly licensed.</p> <p>School computers and networks are vulnerable to viruses. All users of schools systems should be vigilant to reduce the chance of a virus infection.</p>
<p>1.2 Understand the importance of proper hardware maintenance and where applicable be able to perform cleaning and maintenance procedures on hardware</p> <ul style="list-style-type: none">• understand the importance of appropriate storage media care and maintenance• understand the mechanism of a Local Area Network, including servers and clients• be knowledgeable of state of the art technology for micro-computers in the areas of graphics, sound and multimedia capabilities.	<p>There are risks involved in student participation in hardware maintenance. A school situation may make student involvement undesirable. There can be many things for students to learn from these types of responsibilities. Consider having a student assistant. (PSVS)</p>

Unit 2: Problem Solving - Advanced

Suggested Time: 3-4 hours

Foundational Objective

- To provide the students with an opportunity to apply problem-solving methods and techniques in a different context.

Common Essential Learnings Foundational Objective

- To understand the role that humans play in critical and creative thinking. (CCT)

Note: other CELs should be emphasized here.

	Learning Objectives	Notes
2.1	Reorganize a problem so it can be solved in steps.	These objectives have been covered in Computer Science 20 Unit 2 in a different format. A teacher may choose to introduce them with a short review and integrate the content throughout the programming projects in Unit 4. The skills should be continually emphasized with students throughout the course.
2.2	Describe a problem covering the following steps: <ul style="list-style-type: none">• describe the problem state• describe the solution state• distinguish between the two states• identify factors which limit the solutions available• reject extraneous details• apply external knowledge to the solution. (COM)	
2.3	Conceive a model that represents the problem.	There are many options for an appropriate model describing the problem. <ul style="list-style-type: none">• none necessary• picture or diagram• chart or table• tree• top-down diagram using stepwise refinement
2.4	Build the model: <ul style="list-style-type: none">• draw the picture or diagram• design and fill in the chart or table• complete the tree• complete the top-down diagram using stepwise refinement techniques.	

Learning Objectives**Notes**

2.5 Outline a process for solving the problem.

Students should select an appropriate representation of the solution algorithm:

- plain language
- flowcharts
- structured line diagrams.

Build the algorithm:

- write out the instructions (plain language or pseudocode)
- design and fill in the flowchart
- design and fill in the structured line diagram.

2.6 Evaluate the solution use the following steps:

- select appropriate conditions for testing the solution algorithm
- anticipate errors
- test the solution algorithm
- correct errors
- identify limitations of the solution algorithm.

(CCT)

Students often find it valuable to have another student examine their problem solution. The writer of an algorithm may find that a discussion of it helps understanding.

Unit 3: Problem Solving and Programming

Suggested Time: 3-4 hours

Foundational Objective

- To be familiar with and understand the methodology of problem solving with computers.

Common Essential Learnings Foundational Objective

- To understand how knowledge is constructed and evaluated in computer science. (CCT)

Note: other CELs should be emphasized here.

	Learning Objectives	Notes
3.1	Demonstrate an understanding by describing a problem appropriately before beginning a program.	Students should consider: <ul style="list-style-type: none">• problem and solution states• limiting factors• extraneous details• required external knowledge.
3.2	Model the problem appropriately before beginning a program.	Select a model. Build a model.
3.3	Solve the problem appropriately before beginning a program: <ul style="list-style-type: none">• select an algorithm• compose the algorithm• compose a computer program.	
3.4	Re-evaluate the solution by looking back.	Students should: <ul style="list-style-type: none">• select appropriate data for testing• anticipate errors• test and debugidentify limitations of the program.
3.5	Design a program according to structured form containing the parts necessary to the language being used.	Assess and evaluate the design.

Learning Objectives	Notes
3.6 Differentiate between local and global variables and identify the scope of a variable name.	Hold a class discussion.
3.7 Describe the way in which the language utilizes modules that are parts of a main program and explain how data is transferred between different parts of a program. (COM)	Have students do this orally or in writing.
3.8 Demonstrate efficient program development practices.	<p>All of the following steps should be demonstrable.</p> <ol style="list-style-type: none"> 1. Realize the importance of a test whenever a new step is accomplished. 2. Design the main program <ul style="list-style-type: none"> • sequence modules correctly • describe data to be transferred to/from modules. 3. Write stub procedures using built-in traces. A stub procedure is a dummy procedure that is put in place to show that a program structure is working. For example, if a program was written with a procedure to calculate a percentage interest of a sum of money, a stub procedure could return a message of "Interest Procedure is Here". 4. Write internal documentation. 5. Choose data structures. 6. Compose input procedure(s) by designing input screens and data-verification routines. 7. Compose output procedure(s) by designing output screens. 8. Complete and test procedures individually; select appropriate control structures. 9. Test and debug the entire program.
3.9 Compare high-level and low-level programming languages. (CCT)	<p>Name the advantages and disadvantages of:</p> <ul style="list-style-type: none"> • high-level languages • low-level languages.
3.10 Compare interpreted and compiled programming languages.	<p>Name the advantages and disadvantages of:</p> <ul style="list-style-type: none"> • interpreted languages • compiled languages.

3.11 Demonstrate an understanding of the characteristics of **Object-Oriented Programming**.

If an object-oriented language is not being used, this objective should be covered, perhaps through a lecture or demonstration.

The following ideas need to be considered:

- objects, messages and methods
- events and event handling
- Rapid Application Development Systems (RADS)
- Graphical User Interfaces (GUIs).

Unit 4: Experience with Programming and Design - Advanced

Suggested Time: 50-65 hours

Foundational Objective

- To provide hands-on experience with program writing, software design and documentation writing by writing computer programs that will do the processes described.

Common Essential Learnings Foundational Objective

- To apply mathematical and design concepts when writing computer programs. (NUM, COM)

Note: other CELs should be emphasized here.

Learning Objectives	Notes
4.1 Demonstrate an understanding of the way that programs work with the language of the course including: <ul style="list-style-type: none">• basic components of a program• systemic differences between the current language and the one used in Computer Science 20• different parts of the language system; i.e. compiler, editor etc.• internal documentation and other vocabulary.	In most cases, the students will demonstrate their understanding of the concepts and processes through programming projects. An instructor may choose to combine several concepts within a project. It may be appropriate to present a programming problem and introduce the appropriate techniques as tools to help in the solution.
4.2 Demonstrate an understanding of the variable types in the language including: <ul style="list-style-type: none">• different variable types, their uses and the operators associated with each• ways in which values are assigned to variables• acceptable variable names and reserved names. (NUM)	It will help students understand the variable types in the new language if they are compared to the ones in the CS 20 language. Differences and similarities in the functions and reserved names should also be identified.
4.3 Demonstrate an understanding of the expressions and functions of the language, including: <ul style="list-style-type: none">• user-defined functions• function libraries.	

Learning Objectives

Notes

- 4.4 Write documentation for programs. (COM)
- Three types of documents could be considered.
Write a user's manual:
- operating instructions
 - limitations
 - troubleshooting suggestions
- Write supplementary materials (if appropriate):
- workbook
 - activity guide
 - tutorial
- Write a programmer's manual:
- outline of the development process
 - copies of the model and algorithm
 - instructions for modification
- (Include proper internal documentation, remarks, in all programs.)
- Peer testing and evaluation of programs and documentation is very valuable. Often the person who writes a program will be surprised at “obvious” things that new users will miss, and the “crashes” they may cause.
- Bringing in testers from another class can be an interesting variation of this approach.
- 4.5 Demonstrate facility with the input and output processes of the language.
- Good formatting style and proper prompting and error checking must be assessed.
- 4.6 Demonstrate an understanding of top-down program design and the use of modules or sub-programs within programs.
- A student assignment may be to make procedures to fit a program body provided by the instructor. A more rigorous approach would require the students to leave the program body untouched, only adding procedures.
- 4.7 Demonstrate an understanding of decision making and recursion in programs.
- Boolean expressions and logic must be used.
- 4.8 Demonstrate understanding of and facility with single and multi-dimensional arrays with data structures, including searching and sorting.
- The **record** variable type should be introduced during the study of arrays, if the language supports it.

-
- 4.9 Demonstrate understanding of the use of files from within a program including:
- writing to a file
 - reading from a file
 - appending data to a file
 - proper opening and closing of files.
- Individual assessment using a checklist would be possible.

Unit 5: Number Systems and Codes (Optional)

Suggested Time: 5-8 hours

Foundational Objective

- To provide an opportunity for students to learn about the number systems and codes that are fundamental to computer processing of information.

Common Essential Learnings Foundational Objective

- To understand the meaning of numerical systems and the need for precision. (NUM)

Note: other CELs should be emphasized here.

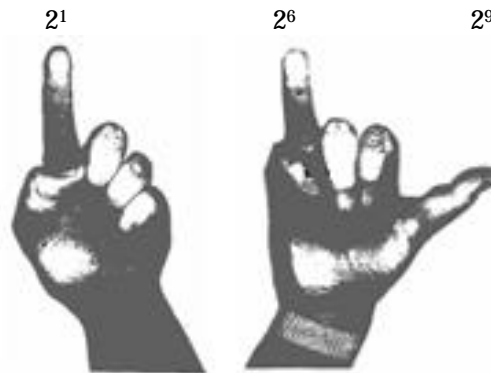
Learning Objectives	Notes
5.1 Demonstrate an understanding of the ASCII code, as follows: <ul style="list-style-type: none">• be able to determine ASCII values of alphabetic and numeric characters• identify the ASCII values of a selection of other common characters and control codes• describe the process that a language uses at the machine level to output a particular character.	An activity where students discover the characters that match the ASCII code numbers through the direct use of the CHR\$() (or the language equivalent) function can be a good way to introduce this topic. It should not take students long to figure out that a program will do the task for them more easily. Beware of low numbers; any below 13 will not return anything.
5.2 Use ASCII codes appropriately in a program, as follows: <ul style="list-style-type: none">• utilize functions in a program to put characters using ASCII• utilize functions in a program to identify the ASCII values of characters• create a program that will sort a list of words into alphabetical order.	An assessment of the program should reveal these codes.

Learning Objectives

Notes

- 5.3 Demonstrate an understanding of the base two number system as follows: (NUM)
- identify the digits (bits) used in binary
 - explain why binary is the number system used in electronic computers
 - using manipulatives or paper, demonstrate counting and simple addition in base two
 - explain that the place values of a binary number are successive powers of two
 - convert values from decimal to binary representation
 - convert values from binary to decimal representation
 - segment and name binary quantities in order to differentiate among the terms bit, nibble, byte and word.

Counting binary on fingers is a good introduction to the base 2 number system. To do this, with palms towards the face, each finger is given a place value. The left thumb = 2^0 (1), the left index finger = 2^1 (2), the left middle finger = 2^2 (4), the left ring finger = 2^3 (8), right thumb = 2^9 (512). When a finger is bent its value is 0; when it is straight, it has the exponential value. The example below shows 578.



$$2 + 64 + 512 = 578$$

- 5.4 Demonstrate an understanding of the base sixteen number system as follows:
- identify the digits (“hexits”) used in hexadecimal
 - explain hexadecimal is used to represent binary values
 - explain that the place values of a hexadecimal number are successive powers of sixteen
 - convert values from decimal to hexadecimal representation
 - convert values from hexadecimal to decimal representation
 - describe the octal number system and how it relates to hexadecimal. (NUM)

The hexadecimal number system is useful in computer science because a 2 digit (“hexit”) hex number represents a byte. The first 4 bits are the first hexit and the last 4 bits are the second.

- 5.5 Create a program that will input values in decimal, binary or hexadecimal format and output the value in the other two. (NUM)

Unit 6: Impact of Information Technology

Suggested Time: 3-5 hours

Foundational Objective

- To provide students with an opportunity to explore the nature of computer science and the societal impact of information technology.

Common Essential Learnings Foundational Objective

- To understand educational empowerment abilities of information technology and its role in supporting a democracy. (PSVS)

Note: other CELs should be emphasized.

Learning Objectives	Notes
6.1 Describe the nature and growth of computer science.	Students need to know about the history of hardware and software development. There are many interesting stories and case studies from the past, for example: <ul style="list-style-type: none">• Conrad Zuse, who built a computer out of telephone switches in his parents living room, is still alive.• Wozniak and Jobs built the first Apple in their garage.• IBM went to Gates and Allen because the CPM people went sailing that day.
6.2 Describe the growing importance of information, comparing industrial and post-industrial societies.	Students will be familiar with the role of information as a commodity. These topics may have been covered in other courses; but there are many opportunities for integration with other subjects. Discuss this with your colleagues. Search the Evergreen curriculum.
6.3 Appraise the economic effects of computer applications.	Students need to be familiar with the effects on: <ul style="list-style-type: none">• transportation• health care• business and industry• the service industry• communications• the military.

6.4 Appraise the effects of computer technology on society. (PSVS, COM)

Exploration of this topic may provide opportunities for guests to visit the class and exchange information and opinions with students.

Students need to be familiar with the effects on:

- employment and careers
- privacy
- interpersonal relationships and communication
- depersonalization
- crime
- dependency on computers.

Small group discussions followed by oral reports and a class discussion is a strategy to use.

Learning Objectives**Notes**

6.5 Appraise the political effects of computer technology.

Students need to be familiar with the effects on:

- the democratic process
- accountability of governments
- international relations.

6.6 Defend personal positions on the appropriate role of computer technology in future society.

Student debates can be a good vehicle for exploring this learning objective.

It might be interesting to do an online discussion, perhaps with expert(s) or another group of students.

Unit 7: Programming for Applications

Suggested Time: 3-4 hours

Foundational Objective

- To provide an opportunity for students to use their programming expertise in an application program.

Common Essential Learnings Foundational Objective

- To value learning for its own sake and as a means to other ends. (IL)

Note: other CELs should be emphasized here.

	Learning Objectives	Notes
7.1	Program and modify macros or programs within an application such as a word processor, spreadsheet or database.	<p>Some languages allow the programmer to access the application from within a program.</p> <p>After computer science students have developed some proficiency with application macros they could tutor students or teachers in macro use. This would raise the level of expertise in the school, be good experience for the trainers, and promote good public relations for the computer program.</p>

Unit 8: Internet and Multimedia (Optional)

Suggested Time: 4-6 hours

Foundational Objective

- To provide an opportunity for the students to increase their understanding of the Internet and to work with multimedia applications.

Common Essential Learnings Foundational Objective

- To use the Internet and multimedia applications to enhance learning situations. (TL)

Note: other CELs should be emphasized here.

	Learning Objectives	Notes
8.1	Demonstrate advanced understanding of Internet functions by: <ul style="list-style-type: none">• sending and receiving e-mail messages with different types of attachments• utilizing the different functions of an Internet search engine• creating (and publishing, if possible) an html page for the WWW.	If these topics are covered adequately in other courses, they may be omitted. In depth work may be appropriate. Students should understand how to narrow or widen a search using Boolean logic.
8.2	Demonstrate increased ability to create multimedia with the use of graphics in a multi-branch presentation.	This unit would complement Communication Production Technology, a Practical and Applied Arts course.

Unit 9: Research Topics

Suggested Time: 7-10 hours

Foundational Objective

- To provide hands-on experience with current research and presentation methods that use information technologies through investigation of a computer science topic of interest.

Common Essential Learnings Foundational Objective

- To outline, research, and present a topic related to computer science. (IL)

Note: other CELs should be emphasized here.

Learning Objectives	Notes
9.1 Research a computer science topic and present the findings to peers using computer technology. (IL, COM)	<p>A list of suggested topics for the student research component follow. Teachers may choose other topics; ones specific to their local community would be fine. The project should make up a substantial part of the coursework, with around 10% of the grade derived from it.</p> <p>Assignments should be made early in the course with the expectation that the students will present their findings to the class. The time allotment for this topic is intended for presentation only. It is important that the students use computer technology of some sort in the presentation of their results. Students should also be encouraged to use technological resources like the Internet in their research.</p> <p>Some possible methods for students to deliver the information included:</p> <ul style="list-style-type: none">• a seminar with technological aides• a multimedia presentation• a series of web pages• a tutorial program• a model.

List of Topics

Architecture of Computing Machines

Explain the organization of the computer being used, at the machine system level.

- Describe the operation of various input devices
- Describe the operation of various output devices
- Explain the role of a microprocessor
- Explain the role and organization of volatile internal memory (RAM)
- Explain the role and organization of non-volatile internal memory (ROM)
- Compare the handling of instructions and data
- Describe the connection of peripherals

Explain the functions of the microprocessor being used. There are many components.

- Accumulator
- Index registers
- Program counter
- Stack pointer
- Other registers
- Stack
- Address bus
- Data bus
- Arithmetic Logic Unit

Recognize the relation between hardware and software at the microprocessor level.

- Interpret a previously-assembled instruction
- Recognize frequently-used instructions from the microprocessor's instruction set

Use basic memory-addressing methods.

- Immediate
- Direct
- Indexed
- Indirect

Explain the features of an assembler.

- Explain the assembly process
- Recognize the fields of an assembly-language instruction

Write simple assembly-language programs.

- Decimal addition and subtraction
- Stack operations
- Branching
- Looping

Demonstrate the ability to save and load assembly-language programs to/from diskette.

Demonstrate the ability to relocate assembly-language programs.

Artificial Intelligence (AI)

Define AI.

Compare theory and practice in AI.

Describe the characteristics of an intelligent machine.

Trace the history of AI.

Describe and compare methods of representing knowledge and logical rules.

Discuss current trends in AI:

- Natural language processing
- Vision
- Speech
- Knowledge representation
- Expert systems
- Uncertain reasoning
- Neural networks

Discuss intelligent robots.

Discuss social issues.

Predict the future of AI.

Career Exploration (Refer to Unit 7 in Computer Science 20)

Field experiences.

- Visit workplaces
- Interview people in various computer-related jobs

Compare expectations in school and in the workplace.

- Determine employer expectations
- Describe employer-employee relationships
- Display attitudes and personal habits which meet employer expectations

Display self-discipline.

- Demonstrate the ability to manage time
- Show a willingness to correct personal weaknesses that limit employability

Participate in a work experience program.

Investigate opportunities for further study.

- Report on offerings by other schools and local business
- Report on offerings by post-secondary institutions (entry requirements, qualifications obtainable, job prospects)

Computer-Aided Design, Drafting, and Manufacturing (CAD/D/M)

Compare the three disciplines:

- commonalties
- differences

Analyze the implications for:

- work environment
- skill requirements
- employment

Describe in detail the parts and functions on one of the disciplines.

Note: see the *Drafting Curriculum Guidelines, A Practical and Applied Art* (online).

Computer Technology

Discuss and/or write about basic solid state electronics.

Design and analyze binary logic circuits.

Describe the operation of power supplies.

Describe the function of logic gates.

Describe the operation of timers, multiplexers, demultiplexers.

Apply Boolean algebra to the design of binary devices.

Construct a single-board computer.

Computing Systems

Operations

- compare batch, interactive, and distributed processing
- contrast various network configurations
- describe a multiprogramming environment

Visit workplaces that make use of these systems.

- observe
- analyze
- interview
- report

Creative Problem Solving

Describe the creative problem solving process.

Apply the process to real-world problems: work with a group.

Incorporate computers into some of the solutions.

Documentation

Examine and criticize commercial documentation:

- tutorials
- user's guides
- programmer's manuals
- systems manuals

Encourage field experience:

- research the characteristics of a user
- write documentation tailored to a user

Future Studies

Report on the writings of various futurists.

- Discuss the likelihood of various trends
- Criticize the predictions of futurists

Study past writing of earlier futurists.

- Compare prediction with fact

Information Storage and Retrieval with Computers

Identify the components of a computer information system.

- information providers, brokers, hosts, carriers

Distinguish between database and database manager.

Distinguish between database manager and file manager.

Use Boolean algebra to formulate query strings.

Apply knowledge of databases to research for some project not related to computers.

Office Automation

Identify the parts of an automated office and describe their function:

- shared applications, shared files, electronic massaging
- roles within the system
- skills required

Discuss security issues in the automated office.

Visit an automated office and report.

Predict trends in office automation.

Project

Apply the programming skills gained in the course to a major project.

- work on the components alone
- work with others, dividing the labour.

Process Control

Describe processes that are under computer control, for example :

- automobile engines and braking systems
- manufacturing
- refining
- transportation
- retail sales and ordering systems

Apply knowledge of computerized process control to a private home situation.

Programming Topics: Advanced

Perform advanced operations on Static Data Structures:

- search (hash)
- sort (shell, quick, radix)

Construct dynamic data structures:

- list, tree, file, stack, queue

Perform operations on dynamic data structures:

- add, delete, search, sort, change, merge
- audit trails

Formulate advanced selection structures using:

- complex Boolean algebra

Use recursion:

- identify conceptual differences between recursion and iteration
- apply recursion to computer solutions of problems

Create simulations

- Use modelling
- Use interactive techniques
- Design a computer simulation
- Discuss the advantages of simulating processes and environments
- Discuss applications to business and industry

Study the theory and practice of computer graphics:

- transformations (translation, rotation, scaling)
- windows and viewports
- clipping
- display devices and how they function
- specific storage and display methods of the microcomputer being used
- input devices and how they function
- applications to business and industry
- turtle graphics

Study the theory and practice of computerized statistical analysis.

- explain basic statistical operations
- use statistics software

Maximize printer performance.

- Explain various methods of generating hard copy
- Describe how the printer being used responds to various codes
- Compare the functions of computer firmware, interface card firmware, and printer firmware
- Apply knowledge of printer performance to computer programs and word processing documents

Special Needs and Computers

Collaborate with a teacher of special-needs students.

- Assess needs
- Determine if a computer might be involved
- Develop a program to fill the needs
- Test and modify the program in real situations

Systems Analysis

Describe the elements of system development:

- analyze the current system
- design a new or modified system.
- implement the new or modified system

Explain the processes involved when analysing an existing system:

- conduct a feasibility study
- gather information
- determine the requirements of the new system
- determine criteria for success
- study alternate systems

Explain the processes involved when designing a new or modified system:

- select hardware
- design forms
- set up files
- set up file-maintenance procedures
- determine security procedures
- write programs and documentation

Explain the processes involved when implementing a system:

- design training schedules
- test the system
- plan for conversion to the new system

-
- plan for maintenance of the new system
- Apply the knowledge of systems analysis to a school situation:
- library, office, cafeteria, student shop

Trends and Issues in Computer Technology

Collect information on some new applications of computer-based technology.

Discuss the social implication of computer applications.

- AI, robotics, simulation, business applications, computers and communications, public information systems, data banks, graphics

Collect information on some recent developments in computers.

- RISC chips, and other hardware
- bit-mapped displays, pull-down/pop-up menus, windows, multiprogramming, and other software developments

Appendix A: Suggested Languages

The selection of programming languages is left to the teacher. However, situations may arise where guidance is needed. These are suggestions for beginners. These are good languages that are easy to obtain, cheap, and relatively easy to use.

Suggested Languages

<i>Operating System</i>	<i>MS-DOS</i>	<i>Windows</i>	<i>Macintosh</i>
<i>CS 20</i>	QBasic	QBasic	True Basic
<i>CS 30</i>	Turbo Pascal	Visual Basic	REAL Basic

Consult the Initial List of Recommended Resources for the titles and annotations of instructional materials.