

*Public and Catholic District School Board Writing Partnerships*

## Technological Education

# Course Profile

## **Computer and Information Science**

Grade 12

University/College Preparation

ICS4M

• *for teachers by teachers*

This sample course of study was prepared for teachers to use in meeting local classroom needs, as appropriate. This is not a mandated approach to the teaching of the course. It may be used in its entirety, in part, or adapted.

---

Course Profiles are professional development materials designed to help teachers implement the new Grade 12 secondary school curriculum. These materials were created by writing partnerships of school boards and subject associations. The development of these resources was funded by the Ontario Ministry of Education. This document reflects the views of the developers and not necessarily those of the Ministry. Permission is given to reproduce these materials for any purpose except profit. Teachers are also encouraged to amend, revise, edit, cut, paste, and otherwise adapt this material for educational purposes.

Any references in this document to particular commercial resources, learning materials, equipment, or technology reflect only the opinions of the writers of this sample Course Profile, and do not reflect any official endorsement by the Ministry of Education or by the Partnership of School Boards that supported the production of the document.

© Queen's Printer for Ontario, 2002

## **Acknowledgments**

### **Public School Board Writing Team – Grade 12, Computer and Information Science**

Public Lead Board

Halton District School Board  
Hans van Wijk, Project Manager

Course Profile Writing Team - Public

Mark Richardson, Halton District School Board (Lead Writer)  
Jaye Herbert, Thames Valley District School Board  
Janet Dudek, Lambton Kent District School Board

Local Reviewers

Sandy Graham, University of Waterloo  
Greg Rodrigo, Georgian College  
Chris Stephenson, Association of Computer Studies Educators

### **Catholic School Board Writing Team – Grade 12, Computer and Information Science**

Catholic Lead Board

Dufferin-Peel Catholic District School Board  
Denise Panunte, Project Manager

Course Profile Writing Team - Catholic

Roy Parteno, Dufferin-Peel Catholic District School Board (Lead Writer)  
Eugene Ladna, Dufferin-Peel Catholic District School Board  
Veronica Hsueh, formerly Dufferin-Peel Catholic District School Board

Local Reviewers

Sandy Graham, University of Waterloo  
Rosaria Kalino, Dufferin-Peel Catholic District School Board  
Carmen Leith, Dufferin-Peel Catholic District School Board (retired)  
Greg Rodrigo, Georgian College  
Chris Stephenson, Association of Computer Studies Educators

---

## Course Overview

### Computer and Information Science, University/College Preparation

**Policy Document:** *The Ontario Curriculum, Grades 11 and 12, Technological Education, 2000.*

**Prerequisite:** Computer and Information Science, Grade 11, University/College Preparation

## Course Description

This course helps students use programming and software engineering principles to design and develop algorithms and programs. Students will use software development and diagnostic tools, implement data structures and algorithms, and use file-management techniques in project settings. They will also develop an understanding of the ethics of computer use and the impact of information technology on the community, and will explore post-secondary education and career paths in computer science.

## How This Course Supports the Ontario Catholic School Graduate Expectations

The Computer and Information Science program in the Catholic faith community enables young adults to develop and utilize their gifts and resources in finding solutions that benefit others in ways that model Gospel values. The curriculum focus enables students to be critical thinkers and innovative problem solvers and analyse the use of resources while understanding the implications of technological innovations. Emphasis on process and results ensures students apply skills and knowledge when providing services and recognize our God-given responsibility to respect the dignity and value of the individual and the need to work co-operatively for the good of all. Computer technology has an ever-increasing effect upon society (e.g., the importance of the ethical use of computers in areas such as piracy, privacy, and security; and the importance of a professional code of computing ethics). It is important for young Catholics to reflect upon and examine the potential of technology to positively and negatively affect lives and careers.

## Course Notes

The Grade 12 Computer and Information Science course prepares students for College and University destinations. The combination of theory and practice encourages students to expand their knowledge and skills in the application of the software design life cycle. Problem solving, logic, and design (a curriculum sub-organizer) are integrated in all units. Students explore career paths and identify which career best suits their interests, aptitudes, and expectations in preparation for their post-secondary destination.

The focus of the Grade 11 course was building problem-solving, data-management, and fundamental programming skills. This Grade 12 course focuses on applying data structures, modular programming, and algorithm development to projects and the management of large projects using the software design life cycle.

Projects, directed challenges, and case studies are drawn from a variety of disciplines and workplace situations. They address a wide spectrum of student interests, are free of bias, and provide opportunities to demonstrate achievement of course expectations.

The course is programming-language independent. The teacher chooses the language that best prepares students for their destinations. Teachers may encourage students to apply their knowledge and skills to additional languages that demonstrate procedural and object-oriented paradigms.

---

The following information was recommended during the university level review in regards to object-oriented programming:

Teachers should strongly consider implementing the ICS4M course using an object-oriented approach. Students should program in an object-oriented language, such as Java, C++, or Object-Oriented Turing, and they should learn OO design techniques. This approach better prepares students who are interested in pursuing Computer Science at the postsecondary level.

After they have completed the ICS3M course, students should be comfortable problem solving and designing procedural programs. Since students who take the Grade 12 course show an aptitude and interest in Computer Science, they benefit from learning about a completely different approach to design and programming. The paradigm shift to an object-oriented perspective is much more than an introduction to a different language and syntax. As always, the choice of language is less important than the emphasis of object-oriented design concepts. Teachers need to emphasize the three basic concepts of OO-programming: encapsulation, inheritance, and polymorphism. Students should also learn new planning techniques, such as Unified Modelling Language (UML) diagrams.

The expectations in the Grade 12 course include a basic understanding of object-oriented programming. If teachers are comfortable introducing the basic OO concepts, these expectations would be best served by writing object-oriented programs. Although it is not essential for students to learn OO programming in order to succeed in Computer Science, students will be at an advantage if they learn and practise proper OO design in their secondary school course.

Effective team skills are key for programmers. Students, in teams, work on case studies and demonstrate conflict-resolution, time-management, task-assignment, and communication skills. Computer professionals communicate using programming standards and documentation as well as visual and oral presentations. These skills are integral to all activities.

When students use the Internet as an information source, it is important for teachers to review and emphasize good information-filtering skills. A session with the school library staff may assist students. Teachers using networked environments should consider the use of shared folders or a website to facilitate management, sharing, and distribution of resources. The use of network resources prepares students for postsecondary learning environments.

The final unit is an authentic assessment in which students apply a range of knowledge and skills through two integrated and meaningful tasks: 1) researching the use of information technology and its impact on the community; and 2) a case study in which the software design life cycle (problem definition, analysis, design, implementation, testing, and maintenance) is followed as the project-management model.

### **Units: Titles and Time**

* Unit 1	Designing and Implementing Data Structures	25 hours
Unit 2	Building Software Libraries	18 hours
* Unit 3	Exploring Advanced Algorithms	22 hours
Unit 4	Managing Software Projects	18 hours
Unit 5	Applying Project-Management and Software-Development Skills	27 hours

\* These units are fully developed in this Course Profile.

---

## Unit Overviews

### Unit 1: Designing and Implementing Data Structures

**Time:** 25 hours

#### Unit Description

In this unit, students review and extend their knowledge of data structures while focusing on implementation of projects to create and manipulate these constructs. Students apply fundamental fixed-size data structures (arrays, user-defined data types, records, arrays of records) to solutions to real-life problems and suggest possible implications of data storage on people's lives in light of Canadian law and Catholic teaching. Students use independent study activity to further their mastery of new programming skills in preparation for postsecondary destinations. They also learn to select proper data structures that best match the information and promote program efficiency, code reusability, and maintenance. Students review and reinforce the principles of ergonomics and relate them to the rights of workers. They explore career opportunities in computing and information science related fields.

#### Unit Overview Chart

Cluster	Learning Expectations	Assessment Categories	Focus
1	IC2.04 CGE2c	Knowledge/Understanding Communication	The importance of ergonomics
2	TFV.02, TF1.03, SP1.06 CGE5a, 5e	Thinking/Inquiry Application	Fundamental data structures
3	TFV.02, TFV.03, TF2.02, SP2.02 CGE4a	Knowledge/Understanding Communication Application	Data-management techniques
4	SPV.05, SP2.02, SP2.12 CGE4a	Thinking/Inquiry Application	File use and peer assessment
5	ICV.03, IC3.01 CGE4g	Knowledge/Understanding Communication	Career research
6	TFV.02, SP2.11 CGE4f	Thinking/Inquiry	New programming skills through research
7	SP1.06, SP2.02, SP2.03, SP3.01 CGE7j	Knowledge/Understanding Thinking/Inquiry Communication Application	Application of data structures

### Unit 2: Building Software Libraries

**Time:** 18 hours

#### Unit Description

Students practise the re-use of code by building and sharing code libraries. The libraries are expanded in subsequent units. Students explore the differences between object-oriented and procedural programming as they apply to software libraries. Students also examine library design in the context of file management in network environments. They investigate intellectual property rights and code ownership from a Catholic perspective and the ethics of code re-use by examining and analysing software-licensing agreements.

---

### Unit Overview Chart

Cluster	Learning Expectations	Assessment Categories	Focus
1	TFV.03, TF1.05, TF2.01 CGE2b	Thinking/Inquiry Knowledge/Understanding	Procedural and Object-Oriented Programming: definitions and differences
2	ICV.01, IC1.02, IC1.03 CGE5a, 5e	Communication Knowledge/Understanding Thinking/Inquiry	May I use your code?
3	TF1.02, TF3.01, TF3.02 CGE7i	Knowledge/Understanding Application	Proper housekeeping and software libraries
4	SP1.04, SP2.04, SP2.05, SP2.09, SP2.12, SP2.13, SP3.02 CGE5g	Application Thinking/Inquiry Communication	Sharing a software library

### Unit 3: Exploring Advanced Algorithms

**Time:** 22 hours

#### Unit Description

Students explore alternative algorithms for solving problems. They examine and program solutions to problems similar to those encountered in ICS3M (e.g., binary search or factorials), using new techniques such as recursion. They also plan solutions to more complex problems using industry-standard methodology (e.g., flow charts, pseudocode, structure charts). Students apply advanced algorithms, such as a recursive sort, to develop more efficient solutions to complex programming problems. Strategies for testing and debugging of programs are developed. Students also calculate cost savings generated by using advanced algorithms as an example of using God-given resources wisely.

#### Unit Overview Chart

Cluster	Learning Expectations	Assessment Categories	Focus
1	TFV.04, TF2.03 CGE5a	Thinking/Inquiry Knowledge/Understanding	New solutions for old problems
2	TF1.04, SP2.06, SP2.10 CGE3c	Thinking/Inquiry Application	Applying recursion to simple problems
3	SP1.02, SP2.07 CGE2e	Application Thinking/Inquiry Communication	Planning a solution
4	TF1.06, SP1.07, SP1.08 CGE7i, 7j	Thinking/Inquiry Application	Simple solutions to complex problems

### Unit 4: Managing Software Projects

**Time:** 18 hours

#### Unit Description

Students examine the components of a software project plan and develop a plan, in the context of case studies, without coding a solution. They review the components of the software design life cycle and explore project management and team-building techniques. They also identify the skills that individuals contribute to the skill-set of the team in the building of Christian leadership. Students create a list of questions, pose the questions to a role-playing client, and write a problem definition and analysis report based upon the answers.

### Unit Overview Chart

Cluster	Learning Expectations	Assessment Categories	Focus
1	TFV.01, TF1.01 CGE5a, 5f	Knowledge/ Understanding	Managing a Project Software Design Life Cycle: Who is on the team?
2	SP1.01, SP2.01, IC2.03, IC3.04 CGE4f	Thinking/Inquiry Communication Application	Problem Definition and Analysis -What's the problem?
3	SP1.02, SP1.03 CGE5a, 5e	Thinking/Inquiry Application	Design -What's the plan? Using skills of team members
4	SP1.03, SP2.01, IC3.02, IC3.03 CGE4f	Communication Application	Implementation – Here's the answer
5	SP1.07, SP2.01 CGE4b	Application	Testing/Maintenance - Is it the right answer? Are there changes to make?

### Unit 5: Applying Project-Management and Software-Development Skills

**Time:** 27 hours

#### Unit Description

This unit is a culminating challenge with two concurrent tasks. Students research, prepare, and present a report examining the use of information technology and its impact in the community and on the common good. They work in groups to apply project-management skills, learned in Unit 4, to a case study. They also plan, develop, test, and document a software solution to a given problem (e.g., an inventory control system for a small business, a record system for a volunteer organization, patient records for a veterinary clinic). Students apply complex programming techniques and utilize software libraries.

#### Unit Overview Chart

Cluster	Learning Expectations	Assessment Categories	Focus
1	ICV.01, ICV.02, ICV.04, IC1.01, IC2.01, IC2.02 CGE5d, 7i	Thinking/Inquiry	Information technology and the community
2	SPV.01, SP1.01 CGE5e	Application	Defining and analysing the problem
3	TFV.05, SPV.01, SPV.04, SP1.02, SP1.03, SP1.05, SP1.06 CGE7i	Knowledge/Understanding Application	Making a plan and defining the roles
4	SPV.01, SPV.02, SPV.03, SPV.05, SP1.03, SP1.04, SP2.02, SP2.03, SP2.06, SP2.12, SP3.02, SP3.03 CGE5g	Application Knowledge/Understanding	Creating and testing a solution
5	SPV.01 SP2.08, SP2.09	Thinking/Inquiry Application	Documenting the solution

---

## Teaching/Learning Strategies

A variety of teaching/learning strategies is used, including plans to address theory and practice in both group and individual activities. This course emphasizes individual research, team building, and project management in the application of the software design life cycle. In preparation for postsecondary destinations, students take increasing responsibility for learning.

Teaching strategies include:

*Brainstorming*: expressing initial ideas with neither criticism nor analysis, e.g., problem-solving discussion in the problem definition and analysis phases of the software life cycle;

*Collaborative/Cooperative*: small-group learning providing high levels of engagement and interdependence (e.g., students working as a team to develop components of a computer program);

*Conferencing*: student-to-student and teacher-to-student discussions;

*Software Life Cycle Design Process*: problem-solving approach using a prescribed series of steps;

*Computer-based Tutorials/Exploration Activities*: use of installed and networked resources, open-ended explorations, and computer projectors, allowing students to work as the teacher demonstrates;

*Independent Study*: exploring and researching a topic of interest;

*Programming*: developing software solutions;

*Computer Research*: using on- and off-line resources;

*Report/Presentation*: presenting research topics to the class using electronic media (e.g., PowerPoint, Corel Presentation);

*Conflict Resolution*: resolving differences in an appropriate manner;

*Whole Group Instruction*: teacher-led instruction to introduce new concepts for skill building.

## Assessment & Evaluation of Student Achievement

Seventy per cent of the grade will be based on assessments and evaluations conducted throughout the course. Thirty per cent of the grade will be based on a final evaluation in the form of an examination, performance, essay, and/or other methods of evaluation.

The assessment and evaluation for Unit 5 is included in the final evaluation. The final exam is important for preparing students to write exams in university and college and for summative assessment of student achievement.

Students are given opportunities to demonstrate their highest level of achievement of the expectations in the four achievement categories.

Students are assessed and evaluated using the following strategies:

*Diagnostic*: at the beginning of a term, a unit of study, or whenever information about prior learning is useful, includes:

- unit pre-tests;
- skill inventory.

*Formative*: during learning, ongoing feedback to students of their strengths, weaknesses, and achievement of the expectations, including:

- communication through software documentation and project reports;
- self-assessment and peer assessment rubrics;
- rubrics;
- checklists for programming problems;
- student/teacher conferencing;
- observation;
- quizzes;
- anecdotal comments with suggestions for improvement.

---

*Summative*: at the end of a learning process, including:

- classroom presentations;
- programming on demand, an in-class assignment using paper and online resources;
- unit tests, final exam;
- culminating challenges in the form of assignments and projects evaluated using rubrics.

## **Accommodations**

Teachers should be aware of students who have an IEP (Individual Education Plan) and ensure that the necessary accommodations are in place.

The following accommodation strategies may be used:

- provide adaptive hardware devices (e.g., large screen monitors, larger fonts, special keyboards);
- provide any environmental changes that may assist in mobility and safety in the classroom;
- continue to develop and use glossaries and word lists of key terms and phrases; make use of diagrams, posters, and handouts that support visual strengths;
- conference with the student as her/his own advocate and appropriate school personnel;
- select a case study context familiar to students to ensure better understanding of the requirements (e.g., students may develop a software package for their church or community group);
- provide opportunities for enrichment throughout the course;
- provide a choice of assignment formats and allow extra time as appropriate to needs;
- provide the following as supports to procuring marks on assessments: extended timeline, wordlists, scribing, alternative presentation format, and frequent feedback/dialogue on one-to-one/team level.

## **Resources**

Units in this Course Profile make reference to the use of specific texts, magazines, films, videos, and websites. The teachers need to consult their board policies regarding use of any copyrighted materials. Before reproducing materials for student use from printed publications, teachers need to ensure that their board has a Cancopy licence and that this licence covers the resources they wish to use. Before screening videos/films with their students, teachers need to ensure that their board/school has obtained the appropriate public performance videocassette license from an authorized distributor, e.g., Audio Cine Films Inc. The teachers are reminded that much of the material on the Internet is protected by copyright. The copyright is usually owned by the person or organization that created the work. Reproduction of any work or substantial part of any work from the Internet is not allowed without the permission of the owner.

The URLs for the websites were verified by the writers prior to publication. Given the frequency with which these designations change, teachers should always verify the websites prior to assigning them for student use. There are many web resources available; the priority in the list is to include Canadian sites. The following resources are used in many activities; specific resources are included in the developed units.

Hume, J.N.P. and D.T. Barnard. *Programming: Concepts and Paradigms*. Toronto: Holt Software Associates Inc., 1997. ISBN 0-921598-27-0

Hume, J.N.P. *Introduction to Programming in Turing*. Toronto: Holt Software Associates Inc., 2001. ISBN 0-921598-42-4

---

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4

Litvin, Marie and Gary Litvin. *C++ for You++*. Andover, Ma, USA: Skylight Publishing, 1999. ISBN 0-9654853-9-0

Wright, Peter. *Peter Wright's Beginning Visual Basic 6.0*. Birmingham, UK: Wrox Press, 1998. ISBN 1-861001-05-3

### **Programming Language Information and Resources**

C/C++/C# – <http://www.accu.org/>

Java – <http://www.holtsoft.com> & <http://www.java.utoronto.ca/Resources/Tutorials/javatutorials.html>

Pascal – <http://www.holtsoft.com>

Qbasic – <http://www.qbasic.com/>

Turing – <http://www.holtsoft.com>

Visual Basic – <http://www.dcs.napier.ac.uk/hci/VB50/home.html>

### **Program Planning**

Fowler, Martin. *UML Distilled: Applying the Standard Object Modelling Language*. Addison-Wesley, 1997.

### **Careers and Career Planning**

Government of Ontario Training and Jobs – <http://www.edu.gov.on.ca/eng/training/training.html>

Human Resources Development Canada – <http://www.hrdc-drhc.gc.ca/common/home.shtml>

### **Postsecondary Education**

Canadian Universities and Colleges from Yahoo! –

[http://ca.yahoo.com/Regional/Countries/Canada/Education/Higher\\_Education/Colleges\\_and\\_Universities](http://ca.yahoo.com/Regional/Countries/Canada/Education/Higher_Education/Colleges_and_Universities)

CommuniCAAT Site (annual calendar) – <http://www.ocas.on.ca>

Government of Ontario Postsecondary Site – <http://www.edu.gov.on.ca/eng/general/postsec/postsec.html>

Ontario Universities Application Centre – <http://www.ouac.on.ca/>

### **OSS Considerations**

The Grade 12 Computer and Information Science course can be used to fulfill the requirement for “an additional credit in science [Grade 11 or Grade 12] or technological education credit [Grades 9-12]” (OSS, 1999, p. 9). This course provides students with educational preparation for university and college.

The curriculum emphasizes theory and concrete applications. Teaching/learning strategies and accommodations are selected to meet the needs of each student. Anti-discrimination education, accommodations for exceptional students, career goals/cooperative education, and community partnerships are addressed. These inclusions support the policies in *Ontario Secondary Schools, Grades 9 to 12: Program and Diploma Requirements, 1999*. Career exploration is available with reference to *Choices Into Action: Guidance and Career Education Program Policy for Elementary and Secondary Schools, 1999*.

---

# Coded Expectations, Computer and Information Science, Grade 12, University/College Preparation, ICS4M

## Theory and Foundation

### Overall Expectations

- TFV.01 · describe the steps in the software life cycle (problem definition, analysis, design, implementation, testing, and maintenance);
- TFV.02 · explain data structures and their processing algorithms;
- TFV.03 · analyse a number of programming paradigms;
- TFV.04 · explain the importance of program correctness and efficiency;
- TFV.05 · describe the relationship among hardware, software, and network requirements.

### Specific Expectations

#### Problem Solving, Logic, and Design

- TF1.01 – describe the components of the software life cycle and their importance in project settings;
- TF1.02 – explain the importance of designing reusable code for large software projects;
- TF1.03 – identify similarities and differences among data structures, including arrays, records, and arrays of records, and their applicability to solving programming problems;
- TF1.04 – evaluate the efficiency of different algorithms and their applicability to solving the same programming problem;
- TF1.05 – describe the difference between procedural and object-oriented programming;
- TF1.06 – explain the levels of program correctness: syntax errors, runtime errors, valid data, invalid data, robustness.

#### Programming Concepts

- TF2.01 – describe how procedural and object-oriented programming paradigms can be used to solve different problems;
- TF2.02 – describe how user-defined types and records provide more flexible and powerful ways of handling data;
- TF2.03 – explain how recursion can be used to solve specific kinds of computing problems.

#### Hardware, Interfaces, and Networking Systems

- TF3.01 – explain the role of a network in accessing computer software resources;
- TF3.02 – describe the issues involved in maintaining a software library (e.g., access, backup, version control);
- TF3.03 – relate hardware requirements to user software demands.

## Skills and Processes

### Overall Expectations

- SPV.01 · incorporate the software life cycle in project settings;
- SPV.02 · effectively use software development and diagnostic tools;
- SPV.03 · implement advanced data structures and algorithms;
- SPV.04 · identify on-line and off-line resource materials;
- SPV.05 · use file management techniques in project settings.

---

## Specific Expectations

### Problem Solving, Logic, and Design

- SP1.01 – devise a plan for a large software project (e.g., an accounts receivable or a random walker program), outlining the required activities at each stage of the software life cycle;
- SP1.02 – use industry-standard methodology (e.g., flow chart, pseudocode, structure chart) in the design process;
- SP1.03 – incorporate modularity, software reuse, and maintenance considerations at the design and implementation stages of the project;
- SP1.04 – incorporate appropriate code from shared software libraries into software projects;
- SP1.05 – select appropriate data structures (e.g., arrays, records, arrays of records) for use in projects;
- SP1.06 – design algorithms to incorporate data structures in projects;
- SP1.07 – ensure program correctness by developing a complete suite of test data (valid and invalid data) to eliminate syntax, runtime, and logic errors;
- SP1.08 – use a problem-solving protocol to troubleshoot computer programs.

### Programming Practices

- SP2.01 – use an integrated development environment to create and manage a project;
- SP2.02 – employ user-defined data types and record data types to improve program efficiency;
- SP2.03 – use arrays, records, and arrays of records in different project settings;
- SP2.04 – build and maintain a small software library to facilitate the reuse of code;
- SP2.05 – incorporate appropriate maintenance considerations during the implementation of programs;
- SP2.06 – use recursion in a simple program;
- SP2.07 – compare the effectiveness of several algorithms for solving the same problem;
- SP2.08 – produce comprehensive documentation (e.g., help files, manuals) for a software project;
- SP2.09 – perform peer reviews of internal and external documentation;
- SP2.10 – perform line-by-line walk-throughs of computer programs that include all program structures;
- SP2.11 – use appropriate research and resource materials to independently master new programming skills;
- SP2.12 – effectively critique programs written by others;
- SP2.13 – log error messages and appropriate fixes.

### Hardware, Interfaces, and Networking Systems

- SP3.01 – implement a backup strategy for program files on different media;
- SP3.02 – develop software libraries in project settings;
- SP3.03 – use predefined modules from software libraries to improve productivity.

## Impact and Consequences

### Overall Expectations

- ICV.01 · describe issues related to the ethical use of computers;
- ICV.02 · describe the use of information technology and its impact in the community;
- ICV.03 · identify postsecondary educational opportunities leading to careers in information systems and computer science;
- ICV.04 · explain the importance of employability skills and lifelong learning to information technology careers.

---

## **Specific Expectations**

### **The Ethical Use of Computers**

- IC1.01** – explain the importance of the ethical use of computers in areas such as software piracy, privacy, and security;
- IC1.02** – describe the essential elements of a code of computing ethics and why it is important to have and follow such a code;
- IC1.03** – analyse current media information relating to ethical issues in computing.

### **Effects of Information Technology**

- IC2.01** – describe how local industries, businesses, or community groups are affected by the growing use of information technology to facilitate communication;
- IC2.02** – describe, using presentation software, how local industries, businesses, or community groups use computers to improve efficiency and productivity to serve their clients;
- IC2.03** – evaluate the pros and cons of moving to new hardware and software technologies (e.g., costs, training requirements, compatibility, deployment);
- IC2.04** – use appropriate strategies to avoid potential health and safety problems associated with computer use, such as musculo-skeletal disorders and eye strain.

### **Postsecondary Education, Career Opportunities, and Employability Skills**

- IC3.01** – describe the range of career opportunities in computing and their lifelong learning requirements;
- IC3.02** – produce job descriptions for occupations/professions in computer and information science;
- IC3.03** – demonstrate communication skills (e.g., the ability to provide comprehensive internal documentation and the ability to explain program design and implementation clearly) in a team setting;
- IC3.04** – describe the elements of working effectively in a team environment (e.g., conflict resolution, time management, constructive criticism, task assignment).

---

## Ontario Catholic School Graduate Expectations

The graduate is expected to be:

**A Discerning Believer Formed in the Catholic Faith Community** who

- CGE1a** -illustrates a basic understanding of the **saving story** of our Christian faith;
- CGE1b** -participates in the **sacramental life** of the church and demonstrates an understanding of the centrality of the Eucharist to our Catholic story;
- CGE1c** -actively reflects on **God’s Word** as communicated through the Hebrew and Christian scriptures;
- CGE1d** -develops attitudes and values founded on Catholic **social teaching** and acts to promote social responsibility, human solidarity and the common good;
- CGE1e** -speaks the **language of life**... “recognizing that life is an unearned gift and that a person entrusted with life does not own it but that one is called to protect and cherish it.” (Witnesses to Faith)
- CGE1f** -seeks intimacy with God and celebrates **communion** with God, others and creation through prayer and worship;
- CGE1g** -understands that one’s purpose or **call in life** comes from God and strives to discern and live out this call throughout life’s journey;
- CGE1h** -respects the **faith traditions**, world religions and the life-journeys of **all people of good will**;
- CGE1i** -integrates faith with life;
- CGE1j** -recognizes that “sin, human weakness, conflict and forgiveness are part of the human journey” and that the cross, the ultimate sign of forgiveness is at the heart of **redemption**. (Witnesses to Faith)

**An Effective Communicator** who

- CGE2a** -listens actively and critically to understand and learn in light of gospel values;
- CGE2b** -reads, understands and uses written materials effectively;
- CGE2c** -presents information and ideas clearly and honestly and with sensitivity to others;
- CGE2d** -writes and speaks fluently one or both of Canada’s official languages;
- CGE2e** -uses and integrates the Catholic faith tradition, in the critical analysis of the arts, media, technology and information systems to enhance the quality of life.

**A Reflective and Creative Thinker** who

- CGE3a** -recognizes there is more grace in our world than sin and that hope is essential in facing all challenges;
- CGE3b** -creates, adapts, evaluates new ideas in light of the common good;
- CGE3c** -thinks reflectively and creatively to evaluate situations and solve problems;
- CGE3d** -makes decisions in light of gospel values with an informed moral conscience;
- CGE3e** -adopts a holistic approach to life by integrating learning from various subject areas and experience;
- CGE3f** -examines, evaluates and applies knowledge of interdependent systems (physical, political, ethical, socio-economic and ecological) for the development of a just and compassionate society.

---

**A Self-Directed, Responsible, Life Long Learner** who

- CGE4a** -demonstrates a confident and positive sense of self and respect for the dignity and welfare of others;
- CGE4b** -demonstrates flexibility and adaptability;
- CGE4c** -takes initiative and demonstrates Christian leadership;
- CGE4d** -responds to, manages and constructively influences change in a discerning manner;
- CGE4e** -sets appropriate goals and priorities in school, work and personal life;
- CGE4f** -applies effective communication, decision-making, problem-solving, time and resource management skills;
- CGE4g** -examines and reflects on one's personal values, abilities and aspirations influencing life's choices and opportunities;
- CGE4h** -participates in leisure and fitness activities for a balanced and healthy lifestyle.

**A Collaborative Contributor** who

- CGE5a** -works effectively as an interdependent team member;
- CGE5b** -thinks critically about the meaning and purpose of work;
- CGE5c** -develops one's God-given potential and makes a meaningful contribution to society;
- CGE5d** -finds meaning, dignity, fulfillment and vocation in work which contributes to the common good;
- CGE5e** -respects the rights, responsibilities and contributions of self and others;
- CGE5f** -exercises Christian leadership in the achievement of individual and group goals;
- CGE5g** -achieves excellence, originality, and integrity in one's own work and supports these qualities in the work of others;
- CGE5h** -applies skills for employability, self-employment and entrepreneurship relative to Christian vocation.

**A Caring Family Member** who

- CGE6a** -relates to family members in a loving, compassionate and respectful manner;
- CGE6b** -recognizes human intimacy and sexuality as God given gifts, to be used as the creator intended;
- CGE6c** -values and honours the important role of the family in society;
- CGE6d** -values and nurtures opportunities for family prayer;
- CGE6e** -ministers to the family, school, parish, and wider community through service.

**A Responsible Citizen** who

- CGE7a** -acts morally and legally as a person formed in Catholic traditions;
- CGE7b** -accepts accountability for one's own actions;
- CGE7c** -seeks and grants forgiveness;
- CGE7d** -promotes the sacredness of life;
- CGE7e** -witnesses Catholic social teaching by promoting equality, democracy, and solidarity for a just, peaceful and compassionate society;
- CGE7f** -respects and affirms the diversity and interdependence of the world's peoples and cultures;
- CGE7g** -respects and understands the history, cultural heritage and pluralism of today's contemporary society;
- CGE7h** -exercises the rights and responsibilities of Canadian citizenship;
- CGE7i** -respects the environment and uses resources wisely;
- CGE7j** -contributes to the common good.

---

## Unit 1: Designing and Implementing Data Structures

**Time:** 25 hours

### Unit Description

In this unit, students review and extend their knowledge in data structures while focusing on implementation of projects to create and manipulate data constructs. Students apply fundamental fixed-size data structures (arrays, user-defined data types, records, arrays of records) to solutions to real-life problems and suggest possible implications of data storage on people's lives in light of Canadian law and Catholic teaching. Students use independent study activity to further their mastery of new programming skills in preparation for postsecondary destinations. They also learn to select proper data structures that best match the information and promote program efficiency, code reusability, and maintenance. Students review and reinforce the principles of ergonomics and relate it to the rights of workers. They explore career opportunities in computing and information science related fields.

### Unit Synopsis Chart

Activity	Time	Learning Expectations	Assessment Categories	Tasks
1.1 Ergonomics and Effects on Workers	2.5 hours	IC2.04 CGE2c	Knowledge/ Understanding Communication	Students observe computer ergonomics in the school.
1.2 Reviewing Fundamental Data Structures and Related Algorithms	3 hours	TFV.02, TF1.03, SP1.06 CGE5a, 5e	Thinking/ Inquiry Application	Students review and improve data structure skills.
1.3 User-Defined Data Types and Arrays of Records	3.5 hours	TFV.02, TFV.03, TF2.02, SP2.02 CGE4a	Knowledge/ Understanding Communication Application	Students use user-defined data types or records and arrays of records in programming problems.
1.4 Random File Access	4 hours	SPV.05, SP2.02, SP2.12 CGE4a	Thinking/ Inquiry Application	Students read and write binary files and participate in a peer-assessment exercise.
1.5 Careers Based on Information Storage and Retrieval	3 hours	ICV.03, IC3.01 CGE4g	Knowledge/ Understanding Communication	Students investigate and share information about computer and information-science related careers.
1.6 Independent Mastery of New Programming Skills	3 hours	TFV.02, SP2.11 CGE4f	Thinking/ Inquiry	Students explore new skills based upon data structures.
1.7 User-Defined Data Types and Data Structures Project	6 hours	SP1.06, SP2.02, SP2.03, SP3.01 CGE7j	Knowledge/ Understanding Thinking/ Inquiry Communication Application	Students apply the knowledge and skills they have acquired to a unit-end project.

---

## Activity 1: Ergonomics and Effects on Workers

Time: 2.5 hours

### Description

By observing computer activity in the school, students assess workplace ergonomics standards and recommend improvements to prevent potential health and safety problems associated with computer use.

### Strand(s) & Learning Expectations

#### Ontario Catholic School Graduate Expectations

CGE2c - presents information and ideas clearly and honestly, and with sensitivity to others.

**Strand(s):** Impact and Consequences

#### Specific Expectations

IC2.04 - use appropriate strategies to avoid potential health and safety problems associated with computer use, such as musculo-skeletal disorders and eyestrain.

### Prior Knowledge & Skills

Students:

- can define the term ergonomics and are aware of job-related injuries, especially in computing;
- can use web browsers and have access to the Internet.

### Planning Notes

- Download checklists and evaluation forms from <http://ergo.human.cornell.edu/cutools.html> to evaluate keyboard, workstation, desk, chair and seating evaluation forms, as well as the male and female musculo-skeletal discomfort questionnaire, left and right hand discomfort questionnaires. Prepare one package for each group of 3-4 students depending on the size of the class.
- Gather specific information from articles on the relationship between improper ergonomics and occurrences of musculo-skeletal disorders like carpal tunnel syndrome, other hand and wrist problems, back pain, and eyestrain. (See Resources for useful sites.)
- Gather specific information on guidelines for good ergonomics: proper light source, keyboard format, seating distances from monitors, posture, chair requirements, required breaks, brief hand, arms and neck exercises done on an hourly basis, proper seating heights, etc. See Resources for useful sites.
- Make arrangement for six different locations (preferably within the school) in which students can make observations on existing computing environment settings, e.g., office, vice principal's office, Guidance and Student Services Department, computer labs, teacher workstations, attendance office, etc.
- Prepare a list of fact-finding techniques (see Appendix 1.1.1).

### Teaching/Learning Strategies

- The teacher presents examples of bad posture/ergonomics and their link to musculo-skeletal injuries.
- The teacher facilitates discussion on the severity of disorders, such as carpal tunnel syndrome.
- Students are assigned into groups of three or four. Each group goes to a specific area of the school at the date and time prearranged with each department/office.
- The teacher presents the fact-finding techniques, checklists, evaluation forms, and questionnaires to students and provides one package to each group.

- 
- Students observe particulars appropriate to their assigned form. If a questionnaire is involved, students negotiate the best time to either pick up the filled-in questionnaire or to help the respondent fill in one. **Note:** Students must be instructed to accept “no comment” as a valid answer to any questions, and to respect that people may choose not to respond at all.
  - Students summarize their findings in a report (see requirements in Appendices 1.1.2 and 1.1.4) and make recommendations/suggestions for improvement. Students should be aware of the professionalism checklist (Appendix 1.1.3).
  - As an extension, students present their recommendations to the respective office/departments.

### **Assessment & Evaluation of Student Achievement**

The teacher and students gather assessment information based on specific expectations, including:

- a formative assessment in the form of student checklists, evaluation forms, and questionnaires (see Appendices 1.1.3 and 1.1.4).
- a summative assessment in the form of a written report using a rubric.

### **Accommodations**

- Plan easily accessible locations for observation.
- Provide a variety of alternatives in presenting information.
- Continue to develop a glossary of key words and phrases.

### **Resources**

Casey, Steven. *Set Phasers on Stun and Other True Tales of Design, Technology and Human Error*, 2nd ed. Santa Barbara: Aegean Publishing Company, 1998. ISBN 0 9636178 85

Chartered Institute of Building Service Engineers (CIBSE) – <http://www.cibse.org/>

Ergonomics For Schools – <http://www.ergonomics4schools.com/infot.htm>

Ergonomics Society – <http://www.ergonomics.org.uk/ergonomics.htm>

Guidelines – <http://ergo.human.cornell.edu/MBergo/schoolguide.html#truth%20ergo%20products>

Norman, D.A. *The Design of Everyday Things*. The MIT Press, 1998. ISBN 0262640376

Puzzles on Ergonomics – <http://ergo.human.cornell.edu/cufun.html>

Safety Library – <http://www.safetyinfo.com/safetyinfo/html/guests/aa-g-indexes/t-ergonomics.htm>

Tools – <http://ergo.human.cornell.edu/cutools.html>

VDT Posture Checklist – <http://ergo.human.cornell.edu/CUVDTChecklist.html>

### **Workplace Ergonomics Tools**

Product Evaluations – <http://ergo.human.cornell.edu/ahKEYBOARD.html> & [/ahSEATING.html](http://ergo.human.cornell.edu/ahSEATING.html)

Posture Evaluations – <http://ergo.human.cornell.edu/ahRULA.html> & [/ahREBA.html](http://ergo.human.cornell.edu/ahREBA.html)

## **Activity 2: Reviewing Fundamental Data Structures and Related Algorithms**

**Time:** 3 hours

### **Description**

Students participate in a timed tournament with various stations. The tournament acts as a review of students’ understanding and skills related to the use of one-dimensional and two-dimensional arrays, developed in Grade 11. Students write short practice programs to review array concepts.

---

## **Strand(s) & Learning Expectations**

### **Ontario Catholic School Graduate Expectations**

CGE5a - works effectively as an interdependent team member;

CGE5e - respects the rights, responsibilities, and contributions of self and others.

**Strand(s):** Theory and Foundation, Skills and Processes

### **Overall Expectations**

TFV.02 - explain data structures and their processing algorithms.

### **Specific Expectations**

TF1.03 - identify similarities and differences among data structures, including arrays, records, and arrays of records, and their applicability to solving programming problems;

SP1.06 - design algorithms to incorporate data structures in projects.

## **Prior Knowledge & Skills**

Students:

- can describe the concepts and use of one- and two-dimensional arrays;
- understand the concepts of the bounds, indices, and homogeneity of data type of arrays;
- can populate, process, and output a one-dimensional array using a counted loop;
- can use array data structures to solve problems;
- can use appropriate algorithms to sort and search a one-dimensional array.

## **Planning Notes**

- Prepare a list of all Grade 11 learning objectives regarding the array data structure. (See Appendix 1.2.1.) Photocopy the list for each student or put an electronic copy on an accessible network drive.
- Assess students' strengths and weaknesses on the topic of arrays.
- Prepare tournament questions according to each subtopic under arrays. (See Appendix 1.2.2.) Prepare answers for each of the questions. Group the questions to suit students (e.g., declaration of arrays (one- and two-dimensional), array bounds and indices, populating an array, searching for an element in an array, sorting an array, output elements from an array, debugging, what does this algorithm do, etc.).
- Prepare tournament tally sheets, one for the group score (Appendix 1.2.3) and one for each individual student's score (Appendix 1.2.4).
- Obtain a stopwatch and whistle and prepare the review programming assignment.

## **Teaching/Learning Strategies**

- The teacher presents the list of learning objectives from Grade 11 regarding the array data structure.
- Students review the material and ask questions to clarify and consolidate concepts.
- The teacher allocates areas in the classroom as questioning stations for the tournament.
- The teacher asks for student volunteers to stay at each station with the questions (and answers). Since the purpose of this activity is to review topics covered in Grade 11, the volunteers rotate to each station and cover all the topics that the competing students cover.
- The teacher assigns students to groups of three or four and reminds them of the importance of developing good teamwork skills both for learning and success in life.
- Students come up with a name for their team and number themselves off within the team.
- The teacher distributes one group total score sheet to each group and an individual score tally sheet to each member of each group (Appendices 1.2.3 and 1.2.4).
- Students form new groups according to their numbers (all ones are a group, etc.).

- 
- Each numbered group visits each station. At a station, each student has a turn at answering a question within a time limit (e.g., 30 seconds). If the answer is wrong, the others can seize the opportunity to answer and gain extra points for their home team. At the whistle, each group rotates to the next station.
  - Students write down questions they answered incorrectly and discuss the answers with other students.
  - The teacher explains that the final long whistle signals the end of the tournament. Home teams then reunite and tally the scores of the members. The team with the highest score wins the tournament.
  - The teacher runs a follow-up session to further clarify and/or answer students' questions after the tournament.
  - Students write practice programs to review and reinforce array programming skills.

### **Assessment & Evaluation of Student Achievement**

The teacher and students gather assessment information from the activity in the form of:

- a diagnostic assessment during the tournament at each questioning station;
- a formative assessment of the review programming assignment (see Appendix 1.2.5).

### **Accommodations**

- Provide a vocabulary list for the tournament and add words to the glossary.
- Provide some tournament questions in audio-tape format as well as on paper.
- Allow some students to be stationary at a questioning station as a timekeeper or recorder.
- Provide opportunities for students to further develop game format or add to the proposed one.

### **Resources**

Bennett, B., C. Rolheiser-Bennett, and L. Stevahn. *Cooperative Learning – Where Heart Meets Mind*. Toronto: Education Connections and Washington: Professional Development Association, 1991. ISBN 0-9695388-0-4

Ministry of Education. *Grade 11 Computer and Information Science Course Profile*. Ontario: Queen's Printer, 2001.

Lambert, K.A., D.W. Nance, and T.L. Naps. *Introduction to Computer Science with C++*, Revised Edition. Boston: PWS Publishing Company, 1997 (Chapters 2 and 3). ISBN 0-534-95204-6

Lewis, J. and W. Loftus. *Java Software Solutions: Foundations of Program Design*. MA: Addison-Wesley, 1998 (Chapter 11). ISBN 0-201-57164-1

## **Activity 3: User-Defined Data Types and Arrays of Records**

**Time:** 3.5 hours

### **Description**

Students critically analyse the effectiveness of one-dimensional, two-dimensional, and related arrays. Students then extend their knowledge of effective means of handling data with the creation and application of user-defined data types and arrays of records.

### **Strand(s) & Learning Expectations**

#### **Ontario Catholic School Graduate Expectations**

CGE4a - demonstrates a confident and positive sense of self and respect for the dignity and welfare of others.

---

**Strand(s):** Theory and Foundation, Skills and Processes

**Overall Expectations**

TFV.02 - explain data structures and their processing algorithms;

TFV.03 - analyse a number of programming paradigms.

**Specific Expectations**

TF2.02 - describe how user-defined type and records provide more flexible and powerful ways of handling data;

SP2.02 - employ user-defined data types and record data types to improve program efficiency.

**Prior Knowledge & Skills**

Students:

- can describe the use of one- and two-dimensional arrays;
- can define and use the bounds, indices, and elements of an array;
- can populate, process, and output a one-dimensional array using a counted loop.

**Planning Notes**

- Prepare chart paper and markers for the brainstorming session.
- Prepare four or five real-life database application problems.
- Prepare four empty shoeboxes by placing two dividers inside, so that each box has three slots. Use masking tape to connect them together, forming a long train-like object. Gather masking tape, four differently coloured markers, four differently coloured sticky pads, and four differently shaped erasers. Place a marker in the first slot of each shoebox, place a sticky pad in the second slot, and place an eraser in the third slot.

**Teaching/Learning Strategies**

- The teacher explains the dynamics of a brainstorm session.
- The teacher groups students and distributes chart paper and markers to each group.
- Students brainstorm pros and cons of the use of one-, two-dimensional and related arrays to solve a given set of scenarios (Appendix 1.3.1). Students have five to ten minutes to complete the task.
- The teacher distributes a real-life database problem to each group.
- Using their current knowledge of arrays, students brainstorm array-based solutions and then share their problems and solutions with other groups.
- The teacher asks students to pay attention to the shoeboxes and point out the similarities (same number of slots; same kind of object inside the first, second, and third slot even though the objects have different colours/shape; each box contains three different objects).
- On the sides of the boxes facing students, the teacher numbers the boxes 1 to 4.
- The teacher asks students if the numbers remind them of something (hopefully, the indices of an array).
- The teacher explains the concept of user-defined data types in relation to the shoeboxes.
- The teacher explains the effectiveness of user-defined data types and use of them in an array.
- The teacher explains the use of dot notation to access data members of an element of an array of user-defined types.
- The teacher asks students to rethink a solution to their database problem. Students apply their understanding of user-defined data types in an array structure to solve the problem.
- Students present their revised solutions.

---

## Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information from the activity in the form of:

- a formative assessment during the group brainstorming sessions;
- a summative assessment of solutions using a rubric or rating scale.

## Resources

Arnow, D. and G. Weiss. *Introduction to Programming Using Java: An Object Oriented Approach*. Massachusetts: Addison & Wesley Longman, Inc., 2000. ISBN 0-201-61272-0

## Activity 4: Random File Access

**Time:** 4 hours

### Description

Students review sequential file access and are introduced to random or direct file access. They explore the advantages and disadvantages of both types of file organization and access methods as well as their use. Students practise creating sequential and direct-access data files, editing them, and appending them. Students learn to arrange fields into records and records into files by choosing the most appropriate file association for the type of data. The exercises and peer assessment of the mini-project solution serve as preparatory stages for the unit-end project in Activity 7.

### Strand(s) & Learning Expectations

#### Ontario Catholic School Graduate Expectations

CGE4a - demonstrates a confident and positive sense of self and respect for the dignity and welfare of others.

**Strand(s):** Skills and Processes

#### Overall Expectations

SPV.05 - use file-management techniques in project settings.

#### Specific Expectations

SP2.02 - employ user-defined data types and record data types to improve program efficiency;

SP2.12 - effectively critique programs written by others.

### Prior Knowledge & Skills

Students:

- are able to effectively use one- and multi-dimensional arrays;
- possess knowledge of sequential file access and flow control structures;
- know the algorithms for at least one sorting and searching technique.

### Planning Notes

- Prepare sample code and sample text files to illustrate and review processing of sequential files (opening files for sequential access, reading from open files, writing strings to files, appending files).
- Develop visual aids to illustrate sequential- and random-access file organization.
- Create sample code in the language of choice to introduce random-access file creation and processing.
- Generate a list of short problems for final group work and clear instructions for random-access file creation and processing (creating records, reading records into variables, writing variables to records, adding records, deleting records, sorting records, searching for records).

---

## Teaching/Learning Strategies

- The teacher reviews sequential-access data files, their organization, file techniques, and the advantages and disadvantages of sequential-access data files.
- The teacher provides sample code and a sample file to illustrate processing of sequential files.
- Students create a sequential file by writing 10 names of their friends, each name followed by the friend's age and grade level. Students manipulate the file. They open it for reading, append it with new fields, delete fields, and then output the file to the screen.
- The teacher supplies students with a text data file. They read the data into an array of string type, output data to the screen, and perform simple sorting and searching techniques on the data. The teacher reminds students to use the End of File (EOF) indicator.
- The teacher introduces random (direct) access data files, contrasting the organization of random-access files and sequential-access files. The teacher stresses the concept of identical records and points out that fields of string data type require fixed length. The teacher visually illustrates the two file-access techniques (e.g., as records on a tape) and the data organization in files (e.g., diagram of file-records-fields-bytes-bits). The teacher discusses the advantages and disadvantages of random file access.
- Students practise creating user-defined data types with fixed-length string fields and record their choices in their notebooks. The teacher leads a class discussion of students' examples, including the correctness of their selection of fields, data types, and lengths of string fields.
- The teacher demonstrates:
  - a) creating random access files;
  - b) opening files for random access;
  - c) editing files opened for random access (by reading records into variables, writing variables to records, adding records, and deleting records).
- Students practise user-defined data file creation and management techniques (e.g., students create book records, including author's name, title, place of publication, publisher, publication date, ISBN, price).
- Students create and edit random access files, sort them, and search for user-specified data. Students collaboratively work on parts of the program and combine their solutions in the final stage. They include proper internal documentation.
- Students swap solutions and provide feedback on correctness and efficiency (Appendix 1.4.1).
- The teacher directs students to the media awareness network website to review "A Day in the Life" from the Annual Report of the Privacy Commissioner of Canada.
- Students describe the structure of any five of the suggested files.
- Students read the Decree on the Media of Social Communications, *Inter Mirifica* by Pope Paul VI, and describe, in several paragraphs, how the concerns expressed in *Inter Mirifica* could be applied to the storage of information about individuals. (Appendix 1.4.2)

## Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information from the activity in the form of:

- a formative peer evaluation checklist (Appendix 1.4.1); students provide feedback on the group's solution (correctness, effectiveness, modularity, internal documentation);
- a summative assessment of the completed program.

## Accommodations

- Enhance work on mini-projects by adding extensions (e.g., use of more advanced sorting techniques, utilization of binary file access, use of more comprehensive records, etc.).

---

## Resources

Barnard, D.T., R.C. Holt, and T.L. West. *Data Structures: An Object-Oriented Approach*. Toronto: Holt Software Associates, Inc., 1995. ISBN 0-921598-24-6

A Day in the Life – [www.media-awareness.ca/eng/issues/priv/topics/privday.htm](http://www.media-awareness.ca/eng/issues/priv/topics/privday.htm)

Deitel, Harvey M. and Paul J. Deitel. *C++: How to Program*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 2000. ISBN 0-13-089571-7

Deitel, Harvey M. and Paul J. Deitel. *Java: How to Program*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 2001, pp. 775, 793. ISBN 0-13-034151-7

Deitel, Harvey M., Paul J. Deitel, and T.R. Nieto. *Visual Basic 6.0: How to Program*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 1999, pp. 579, 616. ISBN 0-13-456955-5

Hume, J.N.P. *An Introduction to Programming in Turing*. Toronto: Holt Software Associates, Inc., 2001. ISBN 0-921598-42-4

Input and Output with Streams – <http://www.ibiblio.org/javafaq/course/week10/>

*Inter Mirifica* – [http://www.vatican.va/archive/hist\\_councils/ii\\_vatican\\_council/documents/vat-ii\\_decree\\_19631204\\_inter-mirifica\\_en.html](http://www.vatican.va/archive/hist_councils/ii_vatican_council/documents/vat-ii_decree_19631204_inter-mirifica_en.html)

Sequential Access Data Files – <http://www.minich.com/education/wyo/cplusplus/cplusplusch11/>

Sequential File Access – <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon98/html/vbconusingsequentialfileaccess.asp>

## Activity 5: Careers Based on Information Storage and Retrieval

**Time:** 3 hours

### Description

Students conduct research on existing occupations and emerging careers in computer and information science. They investigate educational opportunities using college and university calendars (printed or online) and other career-based resources and materials. Students explore their chosen or assigned career in a group and present their research to the class in a formal presentation. The presentation also includes discussion of the ethical aspects of career choices in the information science area.

### Strand(s) & Learning Expectations

#### Ontario Catholic School Graduate Expectations

CGE4g - examines and reflects on one's personal values, abilities, and aspirations influencing life's choices and opportunities.

**Strand(s):** Impact and Consequences

#### Overall Expectations

ICV.03 - identify postsecondary educational opportunities leading to careers in information systems and computer science.

#### Specific Expectations

IC3.01 - describe the range of career opportunities in computing and their lifelong learning requirements.

### Prior Knowledge & Skills

Students:

- are familiar with Internet search and research techniques;
- are aware of job advertisements in, for example, newspapers, online job banks, and the Human Resources Development Canada job banks;
- can tie the current activity with careers in computing they explored in Grade 11.

---

## Planning Notes

**Note:** This activity can be researched entirely through the Internet during class or, if Internet access is not available, prepared beforehand both by the teacher and students. Research of print materials can also supplement the Internet work.

- Obtain post-secondary calendars.
- Prepare advertisements related to information storage professions from newspapers, magazines, etc.
- Contact the guidance staff for career brochures and *The INFO Guide*.
- Create groups to work on a chosen or assigned profession prior to the activity.
- Prepare a list of educational institutions and enterprises to be contacted for information regarding computer and information fields, with a focus on information storage and retrieval.
- Provide students with the list of Internet links related to the careers (Appendix 1.5.1).

## Teaching/Learning Strategies

- The teacher reads selected passages from *Laborem Exercens* and *The Catechism of the Catholic Church* (see Resources) to the class.
- The teacher explains the purpose of the activity, ties it to Grade 11 work on careers, and provides an overview. The teacher distributes student worksheets (Appendices 1.5.1 and 1.5.2) and the presentation assessments (Appendices 1.5.4).
- The teacher reminds students of responsible and ethical use of the Internet.
- Students form groups of three to explore information storage and retrieval professions, either through the use of the Internet or use of previously gathered materials.
- The teacher provides a list of professions or approves professions proposed by the class. Choices include: Data Entry Clerk, Computer/Database Programmer, Database Developer, Database Marketer, Internet Database Developer, Database Manager, Information Systems Programmer, Information Systems Developer, Database Administrator, Information Retrieval Professional/Information Analyst, Health/Medical Information Technician, Biomedical Informatics Professional, Library and Information Science Specialist, Electronic Publishing Professional, and Data Administrator.
- Students research educational programs; suggested terms include: Computer Science, Applied Computer Science, Computer Programming, Information Technology Management, Library and Information Science, Information Organization and Retrieval, Biomedical Informatics, Medical/Health Informatics, Information Management, Information Systems, Information Technology, Database Development, Internet Database Development, Database Programming/Development, Database Marketing, Enterprise Database Management, Information Systems Development/Programming/Management, Computer Applications/Systems Programming.
- Students are assigned one or two periods to explore the chosen/assigned profession and prepare their group presentation. Group information includes: (Member 1) detailed description of the profession; related professions; emerging information storage and retrieval occupations; (Member 2) name of a college, university, or other educational institution that offers a program in the profession and its location, a program description, admission requirements, and program of study; (Member 3) job prospects, possible places of employment, job offers, job requirements, and future changes.
- During the final class, students present the information to their peers in a formal 5- to 10-minute presentation. Ideally, every member presents his/her own findings.
- Students use various forms of communication (e.g., chart paper, overhead transparencies, presentation software, Internet links, etc.) for conducting their presentations.
- Students fill out the summary worksheet, focusing on three careers they may pursue (Appendix 1.5.3).

- 
- The teacher provides a summary of the presentations, eliciting from the class and underlining the importance of databases, Internet databases, and speciality field databases (e.g., health care, education). Students are made aware of emerging information storage and retrieval professions and the changes occurring in the field of information storage. The teacher and students discuss the ethical implications of having access to databases.

### **Assessment & Evaluation of Student Achievement**

The teacher and students gather assessment information from the activity in the form of:

- a summative assessment of student achievement in identifying and describing postsecondary opportunities in information storage/retrieval professions, using a rubric (Appendix 1.5.4).

### **Accommodations**

- For enrichment, students explore emerging professions and speculate on future developments.

### **Resources**

Milbrandt, George and Chris Stephenson. *Careers in Computing*. Toronto: Holt Software Associates, Inc., 2000. ISBN 0-921598-37-8

#### **The Holy See on the Human Community and Work**

Encyclical *Laborem Exercens* –

[http://www.vatican.va/holy\\_father/john\\_paul\\_ii/encyclicals/documents/hf\\_jp-ii\\_enc\\_14091981\\_laborem-exercens\\_en.html](http://www.vatican.va/holy_father/john_paul_ii/encyclicals/documents/hf_jp-ii_enc_14091981_laborem-exercens_en.html)

The Human Community – [http://www.vatican.va/archive/catechism/ccc\\_toc.htm](http://www.vatican.va/archive/catechism/ccc_toc.htm)

## **Activity 6: Independent Mastery of New Programming Skills**

**Time:** 3 hours

### **Description**

In preparation for the unit-end project (Activity 7), students research data structures and their algorithms. The purpose of this independent study unit is to enrich students' knowledge and understanding of data structures, especially dynamic ones. Students examine and learn to create and manipulate one or several major data structures, such as linked lists, stacks, queues, and binary trees. In their research, students identify various applications of dynamic data structures. Students may also explore more efficient sorting and searching algorithms and add them to their newly acquired programming skills, which serve as an introduction to Unit 3: Exploring Advanced Algorithms.

### **Strand(s) & Learning Expectations**

#### **Ontario Catholic School Graduate Expectations**

CGE4f - applies effective communication, decision-making, problem-solving, time, and resource management skills.

**Strand(s):** Theory and Foundation, Skills and Processes

#### **Overall Expectations**

TFV.02 - explain data structures and their processing algorithms.

#### **Specific Expectations**

SP2.11 - use appropriate research and resource materials to independently master new programming skills.

---

## Prior Knowledge & Skills

Students:

- possess knowledge of single- and multi-dimensional arrays, and user-defined data type;
- can use selected sorting and searching algorithms.

## Planning Notes

- Prepare an overview of the purpose of the activity and its application in Activity 7.
- Identify a list of dynamic data structures and their possible applications in real-life projects (overhead).
- Introduce students to the concept of dynamic memory allocation.
- Develop a list of projects that utilize user-defined data types and incorporate dynamic data structures.

## Teaching/Learning Strategies

- The teacher discusses the purpose of the activity and its application in the unit-end project.
- Students learn about dynamic data structures and their applications as outlined on the overhead.
- The teacher presents possible projects and asks students to investigate opportunities for other projects.
- The teacher reads a passage on The Human Virtues to the class before the start of the independent activity: “Human virtues are firm attitudes, stable dispositions, habitual perfections of intellect and will that govern our actions, order our passions, and guide our conduct according to reason and faith. They make possible ease, self-mastery, and joy in leading a morally good life. The virtuous man is he who freely practices the good.” (...) “Human virtues acquired by education, by deliberate acts and by a perseverance ever-renewed in repeated efforts are purified and elevated by divine grace. With God’s help, they forge character and give facility in the practice of the good. The virtuous man is happy to practise them.” Source: <http://www.vatican.va/archive/catechism/p3s1c1a7.htm#1>.
- The teacher and students discuss the reading in reference to qualities needed for independent study.
- The teacher distributes Appendix 1.6.1 and Appendix 1.6.2 to students.
- Students’ research should not only be conducted in class, but also after school and during their work on the unit project (Activity 7). The application of dynamic data structures to the project is evaluated.
- The teacher provides focus, guidance, and help to students during their work.
- The teacher collects the worksheets (Appendix 1.6.1) and proposals (Appendix 1.6.2).

## Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information from the activity in the form of:

- a formative assessment from feedback on the Dynamic Data Structures Worksheet, ensuring that students are properly employing data structures;
- a summative assessment of the project proposal (Appendix 1.6.2).

## Resources

Barnard, D.T., R.C. Holt, and T.L. West. *Data Structures: An Object-Oriented Approach*. Toronto: Holt Software Associates, Inc., 1995. ISBN 0-921598-24-6

Deitel, Harvey M. and Paul J. Deitel. *C++: How to Program*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 2000. ISBN 0-13-089571-7

Deitel, Harvey M. and Paul J. Deitel. *Java: How to Program*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 2001. ISBN 0-13-034151-7

Deitel, Harvey M., Paul J. Deitel, and T.R. Nieto. *Visual Basic 6.0: How to Program*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 1999. ISBN 0-13-456955-5

---

Hume, J.N.P. *An Introduction to Programming in Turing*. Toronto: Holt Software Associates, Inc., 2001. ISBN 0-921598-42-4

Hume, J.N.P. and R.C. Holt. *Introduction to Computer Science using the Turing Programming Language*. Toronto: Holt Software Associates, Inc., 1990. ISBN 0-921598-06-8

Hume, Patterson J.N. and Christine Stephenson. *Introduction to Programming in Java*. Toronto: Holt Software Associates, Inc., 2000. ISBN 0-921598-39-4

Hume, Patterson J.N., T.L. West, R.C. Holt, and D.T. Barnard. *Programming Data Structures in Java*. Toronto: Holt Software Associates, Inc., 1999. ISBN 0-921598-31-9

Shaffer, Clifford A. *A Practical Introduction to Data Structures and Algorithm Analysis, Java Edition*. Upper Saddle River, New Jersey: Prentice Hall, Inc., 1998. ISBN 0-13-660911-2

### **General Overview of Data Structures and Algorithms**

Data Structures and Algorithms – <http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/index.html>

### **Dynamic Memory Allocation**

Efficient Use of Memory – <http://www.csi.uottawa.ca/~holte/T26/lecture2.html>

Dynamic Memory Allocation – <http://dekalb.dc.peachnet.edu/~jbenson/csci1302/dynamic/index.htm>

Dynamic Memory Allocation in C – <http://www.d.umn.edu/~gshute/C/dynamic.html>

## **Activity 7: User-Defined Data Types and Data Structures Project**

**Time:** 6 hours

### **Description**

The intent of the unit-end project is to incorporate knowledge and skills gained in Unit 1 and ICS3M in a solution to a problem. Students work on teacher-approved project topics and follow the specifications. Opportunity for expansion and modification, if approved by the teacher, allows for enrichment.

### **Strand(s) & Learning Expectations**

#### **Ontario Catholic School Graduate Expectations**

CGE7j - contributes to the common good.

**Strand(s):** Skills and Processes

#### **Specific Expectations**

SP1.06 - design algorithms to incorporate data structures in projects;

SP2.02 - employ user-defined data types and record data types to improve program efficiency;

SP2.03 - use arrays, records, and arrays of records in different project settings;

SP3.01 - implement a backup strategy for program files on different media.

### **Prior Knowledge & Skills**

Students:

- possess understanding of and can employ arrays, records, user-defined data types, random file access, dynamic data structures, and the algorithms of at least one sorting and one searching technique;
- are able to implement a backup strategy for program files on different media.

---

## Planning Notes

- Approve proposals (Appendix 1.6.2) if projects conform to requirements (Appendix 1.7.1).
- Develop a list of alternate topics for students.

**Note:** The example detailed in Appendix 1.7.1 involves the use of personal data. This presents several problems including issues of freedom and privacy of information. Unless your school has protocols and safeguards in this area, it would be advisable to use one of the other possible problems from the list of suggestions that does not involve personal data about students or develop fictional data.

## Teaching/Learning Strategies

- The teacher addresses problems encountered in work on dynamic data structures from Activity 6.
- The teacher informs students about approved project topics and helps students whose projects do not meet the requirements select the problem they feel most comfortable with.
- The teacher specifies the requirements and instructs students to follow the software development life cycle model: problem analysis, including a requirements specification, algorithm design, program development and validation, software installation, and maintenance). The cyclical nature of software development and the necessity of solution planning, development, and validation are emphasized. Students provide evidence of their systems analysis and algorithm development (e.g., pseudocode or flowchart). (See Appendix 1.7.1 and Appendix 1.7.2 for details.)
- The teacher chooses fragments of *The Catechism of the Catholic Church* (Section II, The Common Good, and Section III, Responsibility and Participation) to reflect on the concept of a responsible citizen and relate to the views of the Church on man's participation in social life (suggested: 1905, 1912, and 1913) at [www.vatican.va/archive/catechism/p3s1c2a2.htm#II](http://www.vatican.va/archive/catechism/p3s1c2a2.htm#II).
- At the end of the activity, the teacher underlines ethical issues concerning the use of databases.

## Assessment & Evaluation of Student Achievement

- Students submit their solution for summative evaluation using a rubric (Appendix 1.7.3).

## Accommodations

- Allow students to present some components orally (e.g., the requirements specification).

## Resources

### Selected Problem-Solving Resources

Hume, J.N.P. *Problem Solving and Programming in Turing*. Toronto: Holt Software Associates, Inc., 1993. ISBN 0-921598-16-5

Kitto, Richard J. *Computers and Problem Solving, A Case Study Approach*. Toronto: McGraw-Hill Ryerson, Ltd., 1989. ISBN 0-07-549826-X

Stephenson, Chris. Teaching Problem Solving and Design

– <http://www.holtsoft.com/chris/PS&Design.pdf>

Hints for Problem Solving – <http://www.cps.enel.ucalgary.ca/judging/hints.htm>

### Coding Standards Resources

Draft Java Coding Standard – <http://g.oswego.edu/dl/html/javaCodingStd.html>

Netscape's Coding Standards for Java – <http://developer.netscape.com/docs/technote/java/codestyle.html>

Recommended C Style and Coding Standards

– <http://www.doc.ic.ac.uk/lab/secondyear/cstyle/cstyle.html>

Visual Basic Programming Standards – <http://www.gui.com.au/jkcoding.htm>

For user-defined data types, random access, and dynamic data structure, see Resources, Activities 3 and 4.

---

## Appendix 1.1.1

### Fact Finding Techniques

These techniques are often used to gather facts regarding an existing system of work and/or situation.

#### 1. Observations

This method may expose features currently overlooked or taken for granted, such as environmental issues, normal levels of supervision and control, flow of work, bottlenecks of workflow, pace and level of normal workload, and peak loads. It may be time consuming but at times may be the only way to collect data. Use of this method depends on the length of the activity and the skill of the observer.

#### 2. Interviews

There are three distinct stages in this method of data collection. The first phase is the planning of the interview, which involves a clear identification of the purpose of the interview, the time needed for the interview, and obtaining proper authorization for doing the interview. The second phase is to carry out the interview, which involves:

- providing the interviewee with a clear introduction and explanation of the purpose of the interview;
- conducting the interview with good manners;
- controlling and redirecting the flow of the interview;
- asking the appropriate questions;
- being attentive to the body language of the interviewee.
- The third phase is to end the interview on time and request for a subsequent meeting if the interview has not been completed within the agreed-upon time limit. Minutes of the meeting should be promptly provided to the interviewee to check for correct understanding and interpretation of the answers provided.

#### 3. Questionnaires

This technique is used when a personal interview is not possible due to time, cost or distance factors. It is more structured and formal than an interview and requires careful planning in formulating a mixture of open-ended and closed unambiguous questions. A carefully planned questionnaire is a useful tool.

#### 4. Background Reading

This technique involves gathering information from literature published by the workplace in question and provides information about the organization without the need to engage any personnel. Useful literature can be administration procedure manuals, training manuals, memoranda, job descriptions/specifications, and sales literature.

#### 5. Special-Purpose Record Keeping

This method is used to gather information that is not routinely available or kept in any manner, such as information on frequency and time interval of an event, volumes, and trends. The most effective ones do not involve extra workload for the person collecting the data.

#### 6. Analysing Documents

This method analyses the formal flow of information in the organization. It provides information on documents being circulated; when a document is created, amended, and deleted; the meaning, size, format, and source of the document; and its filing sequence.

---

## Appendix 1.1.2

### Ergonomics Report Format

1. **Introductory Paragraph:** Students clearly state the purpose of the activity and the report, demonstrating a clear understanding of the value and purpose of good ergonomics in the workplace for workers' health benefits and for increased productivity in the long run.
2. **Methodology:** Students clearly outline the number of members in the group, the actual workplace observed, the time taken to observe and fill out questionnaires, difficulties encountered and how they were overcome, and the overall cooperation of the entire group. Students use the technique of observation and questionnaires to collect facts on the current ergonomic situation in the particular workplace. Students illustrate their understanding and application of the two techniques and, where applicable, add other techniques they deem necessary for collecting data otherwise inaccessible.
3. **Summary of Findings:** Students outline the means used to summarize the data into meaningful statistics. Students describe their findings regarding all observed categories as outlined in the VDT Posture Checklist (see Resources).
4. **Recommendations:** Using literature provided to them in the package, students make recommendations for improving the ergonomic setting of the observed workplace or give reasons to support and maintain the status quo if the ergonomic settings already meet recommended criteria.
5. **Appendix A:** Students return all completed checklists, evaluation forms, and questionnaires.
6. **Appendix B:** Students return the professionalism checklist after it is filled out by personnel.

## Appendix 1.1.3

### Professionalism Checklist (to be filled in by the employee at the workplace being observed)

Item	Yes	No
Students arrived on time.		
Students had proper attire (e.g., full uniform where applicable).		
Students observed without interrupting daily work routine.		
Students asked questions at appropriate times and rescheduled for another moment if inappropriate.		
Students asked questions courteously.		
Students made good use of the time.		
Students left at the agreed-upon time.		

## Appendix 1.1.4

### Ergonomics Report Checklist

Item	Yes	No
Students provide a clear explanation of the purpose of good ergonomics in the workplace.		
Students describe the value of observation and fact-finding in relation to ergonomics.		
Students have completed and returned checklists, evaluation forms, and questionnaires.		
Students summarize their findings.		
Students analyse the information provided by the various tools.		

---

## Appendix 1.2.1

### Grade 11 Expectations – Arrays

TFV.03 - explain standard control and data structures used in computer programs;

TF2.05 - define the structure of one- and two-dimensional arrays and associated concepts (e.g., subscripts, elements, and bounds);

SP1.03 - select suitable data structures to represent information;

SP1.07 - solve the same problem using various tools (e.g., a calculator and a computer program, a sort program and a spreadsheet/database/word processor sort function);

SP2.02 - incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.03 - write programs that use related arrays to store and extract data;

SP2.10 - incorporate and maintain internal documentation to a specific set of standards;

SP2.14 - trace program execution using manual methods and software debugging tools;

SP2.15 - identify and correct logic, runtime, and syntax errors in programs;

SP2.16 - use linear searches and simple sort routines in programs.

## Appendix 1.2.2

### Tournament Questions and Answers

TFV.03 – explain standard control and data structures used in computer programs.

#### Sample Questions

- Give students a piece of code or pseudocode that uses a counted loop to populate an array and ask students to explain what the code or pseudocode is doing.
- Give students problems to solve that involve the use of repetition, ‘if’ statements, and arrays. Allow students to use only pseudocode (e.g., input 10 students’ names and marks; calculate the average mark; compare each student’s mark with the average; output whether the student is above average or not).

TF2.05 – define the structure of one- and two-dimensional arrays and associated concepts (e.g., subscripts, elements, and bounds).

#### Sample Question

- Display a visual representation of an array. Ask for specific elements at specific index positions, ask for the index position of an element in the array, or ask for an element that is out of bounds of the array (e.g., an array of letters called scrabble):

Index	1	2	3	4	5	6	7	8	9
Element	A	D	T	S	E	N	O	H	L

#### Examples

- Ask students to spell a word using array indices (e.g., EAT is `scrabble(5)+ scrabble(1) + scrabble(3)`).
- Students find the missing letters of a word from the array. They form the word by giving the correct index position of each letter taken from the array (e.g., `_ R _ _ _ A _`; answer 1, 4, 5, 6, 9 to form the word ARSENAL).
- Ask for the letter at index 10 or index 0 to test students’ knowledge of array bounds.
- Provide a number and ask students to form a word out of the index positions that add up to the number (e.g., 11 can be formed by `2+1+3+5`; with these indices, the word DATE is formed).

---

### Appendix 1.2.3

#### Group Total Score Tally Sheet

Group Name:

Members' Names	Individual Score
<b>Total Score:</b>	

### Appendix 1.2.4

#### Individual Score Tally Sheet

Name:

Station	Score
Explain and Define (use of counted loop, the structure of an array and its usage)	
One-dimensional Array (declaration, population, bounds, indices, elements, suitability to problem, suggesting alternate solution with other tools)	
Two-dimensional Array (declaration, population, bounds, indices, elements, suitability to problem, suggesting alternate solution with other tools)	
Related Arrays (usage, application to problems)	
Sorting Algorithms	
Searching Algorithms	
Manual Walkthrough	
Debugging	
<b>Total Score:</b>	

### Appendix 1.2.5

#### A Formative Assessment Programming Assignment

Choose an appropriate data structure (one-dimensional array, related lists of multiple one-dimensional arrays, or multi-dimensional arrays) to solve each of the following questions:

- Calculate the midterm average of a class of 20 students
- Print out the top three students of a class of 20 students, showing names, student IDs, and grades.
- Your teacher wants to let students write a test as many times as they wish. The mark that the student receives is the average mark of all the attempts. Design a system that keeps track of the student name, the number of attempts at the test, and the marks obtained at every attempt, so the average can be calculated at the end.

---

### Appendix 1.3.1

#### Array Storage Scenarios

**Scenario 1:** Maintain a list of site names and corresponding URLs (e.g., Ontario Ministry of Education, <http://edu.gov.on.ca>).

**Scenario 2:** Maintain a list of sites and numbers of hits (e.g., Ontario Ministry of Education, 425 000).

**Scenario 3:** Maintain a list of site names, URLs, and numbers of hits (e.g., Ontario Ministry of Education, <http://edu.gov.on>, 425 000).

### Appendix 1.3.2

#### Presentation Checklist

Item	Yes	No
The solution includes a user-defined data type.		
The user-defined data type meets the requirements of the problem.		
The use of the user-defined data type reflects a clear understanding of the concept.		
The problem is better solved with the user-defined data type.		
There is an array of the user-defined data type.		
The dimension of the array meets the requirements of the problem.		
The size of each dimension of the array is appropriate for meeting the problem requirements.		
The array is populated correctly using a counted loop with the user-defined data type.		
The elements of the array are accessed correctly with the use of array indices.		
The members of each array element are accessed correctly with dot notation and the index.		
The use of the array reflects a clear understanding of the concept of arrays of records.		
The problem is better solved with an array of a user-defined data type.		

### Appendix 1.4.1

#### Mini-Project Peer Assessment Checklist

**Project Title:**

**Group Members:**

Task	Needs Improvement	Comments
1. Proper structure of user-defined data type		
2. Appropriate use of field data types		
3. Effective use of string field lengths		
4. Efficient and clear code development		
5. Proper code modularization		
6. Properly working code for:		
a) creating records		
b) reading records into variables		
c) writing variables to records		
d) adding records		
e) deleting records		
7. Naming convention followed		
8. Brief and informative internal documentation		

---

## Appendix 1.4.2

### Decree on the Media of Social Communications

*Inter Mirifica*, a Decree on the Media of Social Communications, was written in 1963, long before the mass storage of digital information which exists today. The decree focuses on information about individuals and its handling in the communications media of the day. In several paragraphs describe how the concerns expressed in *Inter Mirifica* could be applied to the storage of information about individuals.

## Appendix 1.5.1

### Internet Career Links

#### 1. Career Links

Career Gateway – [www.edu.gov.on.ca/eng/career/](http://www.edu.gov.on.ca/eng/career/)      Career Cruising – [www.careercruising.com](http://www.careercruising.com)  
Top 100 Internet Sites for Learning and Employment – <http://www.jobboom.com/conseils/top-100.html>  
Bridges – <http://cx.bridges.com/>      Youth Resource Network – [www.youth.gc.ca/](http://www.youth.gc.ca/)  
Mazemaster – [www.mazemaster.on.ca](http://www.mazemaster.on.ca)      WRDSB Guidance/Career Ed – <http://guidance.wrdsb.edu.on.ca/>  
Humber Library/Media Services – [http://www.library.humberc.on.ca/RSC\\_Inet/RSC\\_Lnks/wls.html](http://www.library.humberc.on.ca/RSC_Inet/RSC_Lnks/wls.html)  
Canadian Education on the Web – <http://www.oise.utoronto.ca/~mpress/eduweb/eduweb.html>

#### 2. Job Banks

Human Resources Development Canada Job Bank – <http://jb-ge.hrdc-drhc.gc.ca/>  
Canadian Jobs – [www.kenevacorp.mb.ca](http://www.kenevacorp.mb.ca)      National – [www.ijive.com/canada/jobbanks.htm](http://www.ijive.com/canada/jobbanks.htm)

#### 3. Newspapers and Magazines

Links to Canadian Newspapers and Magazines – <http://www.journalismnet.com/papers/canada.htm>  
<http://www-2.cs.cmu.edu/Unofficial/Canadiana/CA-zines.html>  
Globe and Mail – <http://globecareers.workopolis.com/>      Toronto Star – <http://thestar.workopolis.com/>  
The National Post – [www.nationalpost.com/](http://www.nationalpost.com/)      Toronto Sun  
– [www.fyitoronto.com/torsun.shtml](http://www.fyitoronto.com/torsun.shtml)

#### 4. College and University Links

Association of Canadian Community Colleges (ACCC) – <http://www.accc.ca/>  
Canadian Colleges and Universities – <http://www.mit.edu:8001/people/cdemello/ca.html>  
Canadian Colleges Directory – [http://www.campusaccess.com/campus\\_web/educ/e3coll\\_canc.htm](http://www.campusaccess.com/campus_web/educ/e3coll_canc.htm)  
Directory of Canadian Universities – <http://www.aucc.ca/en/acuindex.html>  
INFO Guide to Ontario Universities – <http://www.ouac.on.ca/info/index.htm>  
Ontario's Colleges Career Path – <http://www.careers.ocas.on.ca/index2.html>

#### 5. Subject Directories (selected resources)

Librarians' Index to the Internet – <http://lii.org/>      Ask Jeeves – <http://askjeeves.com>

#### 6. Search Engines (comprehensive databases)

AltaVista – <http://altavista.com>      Excite – <http://www.excite.com>  
Canadian Search Engines – <http://www.journalismnet.com/canada/searchengines.htm>  
Webcrawler – <http://www.webcrawler.com>      Yahoo – <http://www.yahoo.com>

#### 7. Meta Search Engines (search multiple engines at once)

IxQuick – <http://ixquick.com>      MetaCrawler – <http://metacrawler.com>

#### 8. Search and Internet Use Information

Introduction to Search Engines – <http://www.kcpl.lib.mo.us/search/srchengines.htm>  
Searching the Internet – <http://lii.org/search/file/search>  
Comparing Web Search Tools – [www.hamline.edu/administration/libraries/search/comparisons.html](http://www.hamline.edu/administration/libraries/search/comparisons.html)  
Canadian Strategy to Promote Safe, Wise and Responsible Internet Use – [www.aucc.ca/en/acuindex.html](http://www.aucc.ca/en/acuindex.html)

---

## Appendix 1.5.2

### Student Career Research Worksheet

Occupation:

Date:

Group Members:

<b>Detailed description of the profession</b>	<b>Related professions</b>	<b>New emerging information storage and retrieval occupations</b>
<b>Name of postsecondary educational institutions that offer a program in the profession and their locations</b>	<b>Program description</b>	<b>Admission requirements and program of study</b>
<b>Future job prospects and possible places of employment</b>	<b>Samples of job offers in the profession</b>	<b>Job requirements and future changes</b>

## Appendix 1.5.3

### Presentation Summary Worksheet

Name:

Date:

<b>Occupation description</b>	<b>Educational institutions offering the program</b>	<b>Program description</b>	<b>Admission requirements</b>	<b>Program of study</b>	<b>Job prospects and requirements</b>

## Appendix 1.5.4

### Rubric for Postsecondary Opportunities in Information Storage/Retrieval Professions

#### Group Members:

Categories	Level 1 (50-59%)	Level 2 (60-69%)	Level 3 (70-79%)	Level 4 (80-100%)
<b>Thinking/ Inquiry</b> IC3.01 Selection of research materials	- demonstrates limited ability to select appropriate research resources and identify educational opportunities	- demonstrates moderate ability to select appropriate research resources and identify educational opportunities	- demonstrates considerable ability to select appropriate research resources and identify educational opportunities	- demonstrates thorough ability to select appropriate research resources and identify educational opportunities
Analysis and interpretation of material	- analyses and interprets information with little effectiveness	- analyses and interprets information with moderate effectiveness	- analyses and interprets information with considerable effectiveness	- analyses and interprets information with a high degree of effectiveness
Drawing conclusions	- shows limited ability to identify learning requirements and draw conclusions	- shows some ability to identify learning requirements and draw conclusions	- shows significant ability to identify learning requirements and draw conclusions	- shows a high degree of ability to identify learning requirements and draw conclusions
<b>Communication</b> ICV.03. Clarity and effectiveness	- identifies and communicates information with limited clarity and effectiveness	- identifies and communicates information with moderate clarity and effectiveness	- identifies and communicates information with considerable clarity and effectiveness	- identifies and communicates information with a high degree of clarity, effectiveness, and confidence
Use of communication forms	- demonstrates limited command of the various forms of communication	- demonstrates moderate command of the various forms of communication	- demonstrates considerable command of the various forms of communication	- demonstrates extensive command of the various forms of communication

**Note:** A student whose achievement is below Level 1 (50%) has not met the expectations for this assignment or activity.

---

## Appendix 1.6.1

### Dynamic Data Structures Worksheet

Sources used	Dynamic data structures researched	Possible general applications of data structures	Concepts that posed problems in understanding

Commented Code or Algorithm for Sample Use of Dynamic Data Structures:

## Appendix 1.6.2

### Unit-End Project Proposal

Suggested project (name, title)	Purpose	Proposed fields in a database record	Processing to be conducted on the database	What dynamic data structures are applied to the project and how?

---

## Appendix 1.7.1

### User-Defined Data Types and Data Structures Unit Project

**A. Instruction:** Develop a solution to your proposed problem, which has been approved by or selected with the teacher. Example problem: Develop a database of students in your school. Each record includes multiple fields (e.g., Student ID, Last Name, First Name, Address, Date of Birth, Homeroom, Enrolment Date, Timetable, Courses Completed).

Other possible problems:

1. Database of a sports team or a subject club
2. Database of collectibles
3. Equipment inventory database (e.g., computer equipment inventory)
4. Movie database
5. Database of countries and regions
6. Alumni database
7. Subject database (e.g., database of rocks and minerals in a geology classroom)

**B. Project Requirements:** Each of the projects must include the following:

1. a user-defined data type, records, record data types, arrays, arrays of records, random file access, and a dynamic data structure;
2. a file with initial 20 records;
3. more than five fields in a record;
4. sorting and searching of the database based on the user-specified key (each solution must include at least three different keys);
5. retrieval of records from a file based on a search criterion or location of a record in the file;
6. number of records in a file must be displayed after each run or update of the database;
7. appending the file, updating records, deleting and inserting records.

## Appendix 1.7.2

### Programming Standards and Conventions: Interface, Coding Standards, Documentation, and Program Testing

**A. User Interface:** Interface design and layout are logical, user friendly, easy to follow, readable, and well balanced.

**B. Coding Standards:** Follow proper code-writing conventions developed for the programming language of your solution. Include proper naming and code indentation.

**C. Internal Documentation:** Your code must include internal documentation with a header in the form of a comment at the top. Entries must be aligned properly. The header must include: project title, programmer's name, instructor's name, course code and name, school, software used, hardware used, due date, completion date, and program description. Major structures and segments of the code must be documented with comments indicating their role in the program.

**D. Program Testing:** Test your program thoroughly. Test your solution for extreme data and other data types. Anticipate entries from the user. Ask your peers to test your program before submission.

## Appendix 1.7.3

### Rubric for User-Defined Data Types and Data Structure Project

Categories	Level 1 (50-59%)	Level 2 (60-69%)	Level 3 (70-79%)	Level 4 (80-100%)
<b>Thinking/Inquiry</b>	<b>The student:</b>			
SP1.06 - design algorithms to incorporate data structures in projects	- designs algorithms with limited application of data structures	- designs algorithms with moderate application of data structures	- designs algorithms with considerable application of data structures	- designs algorithms with a thorough application of data structures
<b>Application</b>	<b>The student:</b>			
SP2.02 (Programming Practices) - employ user-defined data types and record data types to improve program efficiency	- employs user-defined data types and record data types to improve program efficiency with limited effectiveness	- employs user-defined data types and record data types to improve program efficiency with moderate effectiveness	- employs user-defined data types and record data types to improve program efficiency with considerable effectiveness	- employs user-defined data types and record data types to improve program efficiency with a high degree of effectiveness
SP2.03 (Programming Practices) - use arrays, records, and arrays of records in different project settings	- employs arrays, records, and arrays of records in different project settings with limited effectiveness	- employs arrays, records, and arrays of records in different project settings with moderate effectiveness	- employs arrays, records, and arrays of records in different project settings with considerable effectiveness	- employs arrays, records, and arrays of records in different project settings with a high degree of effectiveness
SP3.01 (Hardware, Interfaces, and Networking Systems) - implement a backup strategy for program files on different media	- implements backup strategy on media with limited effectiveness	- implements backup strategy on media with some effectiveness	- implements backup strategy on media with considerable effectiveness	- implements backup strategy on media with a high degree of effectiveness
Programming Standards and Conventions - user interface, coding standards, internal documentation, program testing	- follows programming standards and conventions to a limited degree	- follows programming standards and conventions to some degree	- follows programming standards and conventions to a considerable degree	- follows programming standards and conventions to a high degree

**Note:** A student whose achievement is below Level 1 (50%) has not met the expectations for this assignment or activity.

---

## Unit 3: Exploring Advanced Algorithms

**Time:** 22 hours

### Unit Description

Students explore alternative algorithms for solving problems. They examine and program solutions to problems similar to those encountered in ICS3M (e.g., binary search or factorials), using new techniques such as recursion. They also plan solutions to more complex problems using industry-standard methodology (e.g., flow charts, pseudocode, structure charts). Students apply advanced algorithms, such as a recursive sort, to develop more efficient solutions to complex programming problems. Strategies for testing and debugging of programs are developed. Students also calculate cost savings generated by using advanced algorithms as an example of using God-given resources wisely.

### Unit Synopsis Chart

Activity	Time	Learning Expectations	Assessment Categories	Tasks
3.1 New Solution for Old Problems	5 hours	TFV.04, TF2.03 CGE5a	Thinking/Inquiry Knowledge/ Understanding	Group problem solving, discussion, exercises, assignment
3.2 Applying Recursion to Simple Problems	5 hours	TF1.04, SP2.06 CGE3c	Thinking/Inquiry Application	Discussion, lab exercises, assignment
3.3 Planning a Solution	6 hours	SP1.02, SP2.07 CGE2e	Application Thinking/Inquiry Communication	Group problem solving, writing algorithms, creating flowcharts and pseudocode, communicating solutions
3.4 Simple Solutions to Complex Problems	6 hours	TF1.06, SP1.07, SP1.08, SP2.10 CGE7i, 7j	Thinking/Inquiry Application	Discussion, group problem solving, assignment, unit test

### Activity 1: New Solutions for Old Problems

**Time:** 5 hours

#### Description

Students examine problems that can be solved using more than one algorithm (e.g., determining the factorial value of a number). Some of these problems were solved in ICS3M using a different method. Using brainstorming or other group problem-solving techniques, students develop alternative algorithms using recursive and non-recursive techniques. Students identify the components of a recursive algorithm and develop criteria for recognizing when a recursive algorithm may be applied.

#### Strand(s) & Learning Expectations

**Strand(s):** Theory and Foundation

#### Overall Expectations

TFV.04 - explain the importance of program correctness and efficiency.

#### Specific Expectations

TF2.03 - explain how recursion can be used to solve specific kinds of computing problems.

---

## Prior Knowledge & Skills

Students:

- can use problem-solving models and develop appropriate algorithms to solve problems;
- are able to write pseudocode.

## Planning Notes

- Review the nature of recursion (see Appendix 3.1.1).
- Gather examples of problems that can be solved using more than one method, including recursion, and determine which problems may be solved using a recursive algorithm.

## Teaching/Learning Strategies

- The class is divided into groups of two or three students.
- The teacher reviews the brainstorming problem-solving technique.
- The teacher presents a problem that can be solved using a familiar but complex algorithm and may also be solved using a less familiar but simpler algorithm (e.g., determining the quotient and remainder of the division of two).
- Students, in their groups, develop more than one algorithm for the solution.
- The teacher facilitates a class discussion to develop criteria for the evaluation of algorithms, including the efficiency of the solution and the complexity of the required coding. Both processing and user interface efficiencies are considered.
- Groups evaluate the algorithms using the developed criteria and share their algorithms and evaluations with the class.
- The teacher introduces the recursive method of problem solving and illustrates a recursive algorithm for the solution to a different problem (e.g., calculating the factorial value of a number).
- Groups develop a recursive algorithm to the initial problem and evaluate its efficiency.
- The teacher facilitates a class discussion to establish criteria for determining if a recursive algorithm is an appropriate solution and identifies additional problems that may be solved using recursion.
- Working in groups, students develop recursive and non-recursive algorithms for additional, assigned problems (see Appendix 3.1.2).

## Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations, including:

- a formative assessment of the assigned in-class work in the form of roving conferences;
- a summative assessment in which students complete an assignment requiring the development of both a recursive and a non-recursive algorithm (see Appendix 3.1.3).

## Accommodations

- Provide print copies of examples of algorithms using recursive and non-recursive methods, including graphic illustrations, and use models to illustrate the algorithms.

## Resources

Data Structures Lecture Notes, Recursion – <http://www.csi.uottawa.ca/~holte/T26/lecture3.html>

The Fibonacci Series – <http://library.thinkquest.org/27890/mainIndex.html>

Hume, J.N. Patterson and Christine Stephenson. *Introduction to Programming in Java*. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4

Hume, J.N.P. and D.T. Barnard. *Programming: Concepts and Paradigms*. Toronto: Holt Software Associates Inc., 1997. ISBN 0-921598-27-0

---

## Activity 2: Applying Recursion to Simple Problems

**Time:** 5 hours

### Description

Students work in pairs to write programs, applying the algorithms developed in Activity 1. Using the criteria developed in Activity 1, students evaluate the efficiency of their programs and the appropriateness of the recursive method.

### Strand(s) & Learning Expectations

**Strand(s):** Theory and Foundation, Skills and Processes

#### Specific Expectations

TF1.04 - evaluate the efficiency of different algorithms and their applicability to solving the same programming problem;

SP2.06 - use recursion in simple programs.

### Prior Knowledge & Skills

Students:

- can manipulate data in one- and two-dimensional arrays;
- can incorporate user-defined functions in programs and can write subroutines that pass parameters;
- have knowledge of local and global variables.

### Planning Notes

- Review the application of recursive programming techniques in the chosen programming language.
- Select examples of problems that can be solved by using recursive programming techniques.

### Teaching/Learning Strategies

- Working in pairs, students program a non-recursive solution to a problem introduced in Activity 1 (e.g., finding the quotient and remainder of the division of two numbers) (see Appendix 3.2.1).
- The teacher illustrates the application of recursive programming techniques by demonstrating a simple recursive solution (e.g., finding the factorial value of a number), tracing the execution of the program to illustrate how recursion functions.
- The teacher reviews variable scope, parameter passing, and the criteria for use of user-defined functions and sub-programs.
- Students develop additional programs using the algorithms developed in Activity 1.
- The teacher facilitates a class discussion to develop criteria for evaluating the efficiency of the application of the algorithms and the efficiency of program code (see Appendix 3.2.2).
- Groups evaluate their solutions using the established criteria.
- The teacher introduces the binary search algorithm.
- Students individually complete a programming assignment requiring them to develop a recursive binary search algorithm (e.g., finding a telephone number in a list of names and telephone numbers).

### Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations, including:

- a formative assessment of the assigned in-class work in the form of roving conferences;
- a summative assessment of the programming assignment (see Appendix 3.2.3).

---

## Accommodations

- Provide “scaffolded” programs (i.e., incomplete program listings with subroutine headings and basic structure included) to help struggling students.
- Provide code examples that demonstrate use of recursive sub-programs and functions.
- Challenge students to attempt more complex problems.

## Resources

Hume, J.N. Patterson and Christine Stephenson. *Introduction to Programming in Java*. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4

Hume, J.N.P. and D.T. Barnard. *Programming: Concepts and Paradigms*. Toronto: Holt Software Associates Inc., 1997. ISBN 0-921598-27-0

Programming with Recursions – <http://erwnerve.tripod.com/recursion.htm>

Recursion – <http://et.nmsu.edu/~etti/winter98/computers/jenkins/jenkins.html>

## Activity 3: Planning a Solution

**Time:** 6 hours

### Description

Students work in groups to analyse complex problems (e.g., Towers of Hanoi) and to develop appropriate algorithms using the recursive and non-recursive techniques examined in Activities 1 and 2. Students create flowcharts and pseudocode to assist them in planning a solution and assess these representations of code as problem-solving tools.

### Strand(s) & Learning Expectations

**Strand(s):** Skills and Processes

#### Specific Expectations

SP1.02 - use industry-standard methodology (e.g., flow chart, pseudocode, structure chart) in the design process;

SP2.07 - compare effectiveness of several algorithms for solving the same problem.

### Prior Knowledge & Skills

Students:

- can apply the steps in the software design life cycle;
- have used pseudocode, diagrams, and charts to summarize program design;
- can develop appropriate algorithms to solve problems.

### Planning Notes

- Review the application of top-down problem solving (see Appendix 3.3.1).
- Select a problem to use in developing a model solution (see Appendices 3.3.2, 3.3.3, and 3.3.4) and prepare the appropriate models.

### Teaching & Learning Strategies

- The class is divided into groups of two or three students and each group is assigned a problem.
- Groups investigate the problem, using a variety of problem-solving techniques to analyse it.
- Each group uses brainstorming or other group problem-solving techniques to develop an algorithm for the solution to the problem. In a class discussion, groups present and share their algorithms.

- 
- Students compare the effectiveness and efficiency of the algorithms presented and then the groups refine their algorithms.
  - Each group develops a flow chart, structure chart, and/or pseudocode to represent the application of the algorithm. The teacher conferences with each group to discuss and assess the solution design.

### **Assessment & Evaluation of Student Achievement**

The teacher and students gather assessment information based on specific expectations, including:

- a formative peer assessment of the presented algorithms (see Appendix 3.3.6);
- a formative assessment of the design for the solution to the problem (see Appendix 3.3.7).

### **Accommodations**

- Provide print sample algorithms similar to the one studied.
- Use graphical models to illustrate the problem.
- Selectively pair/group students to assist problem solving.
- Provide problems of varying complexity to provide an appropriate challenge.

### **Resources**

How to Draw Flowcharts – <http://www.smartdraw.com/resources/centers/flowcharts/tutorial1.htm>

Hume, J.N. Patterson and Christine Stephenson. *Introduction to Programming in Java*. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4

Hume, J.N.P. and D.T. Barnard. *Programming: Concepts and Paradigms*. Toronto: Holt Software Associates Inc., 1997. ISBN 0-921598-27-0

Pseudocode Guidelines – <http://www.caam.rice.edu/~caam211/JZoss/pseudocode.html>

Structured Pseudocode – <http://mstack.cs.niu.edu/c360/spring01/notes/StruPseu.html>

## **Activity 4: Simple Solutions to Complex Problems**

**Time:** 6 hours

### **Description**

Students write programs to apply the algorithms developed in Activity 3. They explore differing levels and types of errors, and develop test data and troubleshooting routines to ensure robust programs.

### **Strand(s) & Learning Expectations**

**Strand(s):** Theory and Foundation, Skills and Processes

#### **Specific Expectations**

TF1.06 - explain the levels of program correctness: syntax errors, runtime errors, valid data, invalid data, robustness;

SP1.07 - ensure program correctness by developing a complete suite of test data (valid and invalid data) to eliminate syntax, runtime, and logic errors;

SP1.08 - use a problem-solving protocol to troubleshoot computer programs;

SP2.10 - perform line-by-line walkthroughs of computer programs that include all program structures.

---

## Prior Knowledge & Skills

Students:

- can use recursion in simple programs and trace through a recursive routine;
- can apply industry-standard methodology in designing programs;
- can analyse the effectiveness of different algorithms.

## Planning Notes

- Review notes on errors (see Appendix 3.4.1) and debugging/testing strategies (see Appendix 3.4.2).
- Review the software life cycle and its application to program testing and maintenance.
- Prepare programs with a number of errors for debugging purposes.

## Teaching/Learning Strategies

- Students work in groups to identify the types of errors present in sample programs, including compile-time (errors of syntax or semantic), run-time, and logic errors.
- Using sample programs, students test for robustness and input validity using the black box and glass-box approaches (see Appendix 3.4.2).
- Students use debugging and tracing to review the recursive algorithms developed in Activity 2 and develop a suite of test data to test these algorithms (see Appendix 3.4.3).
- Students write the code for an algorithm developed in Activity 3 and then create a test driver subroutine or method to test their programs for robustness and input validity.
- Students individually complete an assignment (see Appendix 3.4.4) and write a unit test (see Appendix 3.4.6).

## Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations, including:

- a formative assessment of the assigned work as students complete the programming assignment;
- a summative assessment of the programming assignment using a rubric (see Appendix 3.4.5);
- a summative unit test (see Appendix 3.4.6).

## Accommodations

- Use graphical models to illustrate the example program errors.
- Challenge students to attempt more complex problems.

## Resources

Hume, J.N.P. and D.T. Barnard. *Programming: Concepts and Paradigms*. Toronto: Holt Software Associates Inc., 1997. ISBN 0-921598-27-0

Kruse, Robert. *Data Structures & Program Design*. Prentice Hall PTR, 1994.

Schneider, G.M. and S.C. Bruell. *Concepts in Data Structures and Software Development*. Wiley, 1991.

Verifying, Testing, and Debugging – <http://www.cs.und.edu/~jo/cs161/note/02/ppframe.htm>

---

## Appendix 3.1.1

### An Introduction to Recursion

Recursion is not just a computer programmer's problem-solving technique; there are many examples of recursion around us. Consider a TV camera that is focused on a scene that includes a TV monitor displaying the image recorded by the camera. The image seen on the monitor includes an image of the monitor displaying an image of the monitor and so on – that's recursion.

We also see recursion in the "House of Mirrors" at the Fall Fair. Two flat mirrors face each other. Looking into one of the mirrors, we see our own image and that of the mirror behind us, which includes a view of our back and the mirror in front of us, which includes an image of the mirror behind, and so on. There are recursive solutions to everyday problems as well. Faced with the chore of lifting a truckload of shingles to a rooftop, we are presented with two alternatives. Hire a crane to lift them all at once or use a recursive method consisting of:

- sub-divide the pile of shingles into groups until the shingles are divided into groups small enough to be carried – individual shingles if necessary;
- carry a group up the ladder and place it on the roof;
- climb down the ladder;
- pick up another group of shingles;
- carry it up the ladder and place it on the roof.

And so on. Continue until there are no more shingles on the ground, and it is time to carry the tools to the roof to begin installing the shingles.

For a problem to have a recursive solution it must meet the following criteria:

- There must be a way of reducing the problem to smaller, repetitive elements (e.g., dealing with the shingles one bundle at a time rather than the entire load of shingles).
- There must be a way of identifying, recognizing, and dealing with the endpoint of the solution (e.g., there are no more shingles left on the ground so carry the tools to the roof).
- There must be a way of achieving the larger result from the component elements (e.g., the end result of both methods is the entire load of shingles on the roof).

Calculating the factorial value of a number (!) is an example of a problem that may be solved recursively. The factorial value is defined as the product of all natural numbers equal to or less than the number

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040$$

A non-recursive solution for calculating the factorial value may be achieved with a loop structure.

The problem may also be solved with a recursive solution. To satisfy the first criterion, the problem may be reduced to these repetitive elements:

$$7! = 7 \times 6!$$

$$6! = 6 \times 5!$$

$$5! = 5 \times 4!$$

And so on.

The end point of the problem (also known as the base or degenerate case) comes with the definition that  $1! = 1$  and at that point the algorithm has reached its endpoint. This satisfies the second criterion.

Linking each of the smaller problems satisfies the third criterion. The solution to the base case is 1. That allows determining that the second to last case ( $2! = 2 \times 1!$ ) has a value of 2. This value is passed to the preceding case allowing it to have a solution of 6, and so on.

Learning to recognize when a recursive method may be applied in the solution of a problem is an important skill to a programmer.

---

## Appendix 3.1.2

### Simple Problems with Recursive and Non-Recursive Solutions.

1. Your calculator is broken and the divide button no longer works. List the steps that would be required to find the quotient and remainder for the division of two numbers using the broken calculator.
2. A geometric sequence of numbers is created from a “first term” (T1) and a “common ratio” (R). The second term is formed by multiplying the first term times the common ratio, the third term is formed by multiplying the first term times the common ratio squared, the fourth term by multiplying the first term time the cube of the common ratio and so on.  
For example, if the first term is 3 and the common ratio is 2 then:  
 $T_1=3$   
 $T_2=3 \times 2=6$   
 $T_3=3 \times 2^2=12$   
 $T_4=3 \times 2^3=24$   
 $T_5=3 \times 2^4=48$   
And so on.  
Unfortunately, your calculator doesn't have any kind of power function, only buttons labelled +, -,  $\times$ , and / (but that one's broken). Describe at least two methods for calculating the value of a particular term in a geometric series with this calculator.
3. The greatest common divisor is defined as the largest number that divides into two other numbers. List the steps required to determine the greatest common divisor of two given numbers.
4. In the year 1202, a mathematician by the name of Leonardo Fibonacci described a very simple series that has intrigued other mathematicians ever since. The Fibonacci series is:  
1 1 2 3 5 8 13 21 34 55 ...  
Describe a recursive and non-recursive method of determining the value of any specified term of the Fibonacci series.
5. Develop a recursive algorithm to determine if there is a palindrome hidden within a longer word or phrase. A palindrome is a word or phrase that has the same sequence of letters when read from left to right and when read from right to left, ignoring the spaces (e.g., “Some like cake, but I prefer pie” contains the palindrome “I prefer pi”).

---

## Appendix 3.1.3

### Developing Alternative Algorithms Assignment

In 1653, a French mathematician named Blaise Pascal described a triangular arrangement of numbers that is used to determine the probability of certain events. Pascal's Triangle is constructed as follows:

- the first row has only one element and it is 1;
- each successive row contains 2 more elements than the previous row;
- the first and last elements of each row are 1;
- the value of the remaining elements in each row are found by adding together the two elements found in the row above.

The first seven rows of Pascal's triangle are:

					1								
				1		1							
			1		2		1						
		1		3		3		1					
	1		4		6		4		1				
1		1		5		10		10		5		1	
	1		6		15		20		15		6		1

Develop a non-recursive and a recursive algorithm for determining the value of an element in the triangle given the row and element number (e.g., given row 6 and element 5, the value to be found is 10).

#### Assessment Checklist

Task	Demonstrated	Comment
Solution allows for user-selected number of rows.	Y / N	
Solution allows for user-selected element from the row.	Y / N	
<i>Non-Recursive Solution</i>		
Identifies the need for a two-dimensional array.	Y / N	
Correctly initializes the array values appropriate for the inputted data.	Y / N	
Returns the correct value for selected test data.	Y / N	
<i>Recursive Solution</i>		
Correctly identifies sub-program parameters.	Y / N	
Correctly identifies the recursive sequence.	Y / N	
Identifies the base case.	Y / N	
Returns the correct value for selected test data.	Y / N	

---

## Appendix 3.2.1

### Program Code Example

#### (Determining the Quotient and Remainder for the Division of Two Numbers)

##### Non-recursive Solution (Visual Basic)

```
Option explicit
Dim dividend, divisor, quotient, remainder as single
dividend = text1.text
divisor = text2.text
quotient = 0
subtotal = dividend
do
    subtotal = subtotal - divisor
    quotient = quotient + 1
loop until subtotal < divisor
Label1.Caption = "The quotient is " + STR(quotient) + " with remainder " + STR(subtotal)
```

##### Recursive Solution

```
Option Explicit
Private Sub divide(dividend As Single, divisor As Single, quotient As Single, remainder As Single)
If dividend >= divisor Then
    Rem subtract divisor from dividend and increment quotient
    quotient = quotient + 1
    dividend = dividend - divisor
    call divide(dividend, divisor, quotient, remainder)
Else
    Rem base case
    remainder = dividend
End If
End Sub

Private Sub Command1_click()
Dim num1, num2, q, r as single
Rem divide num1 by num 2, returning the divisor and remainder
num1 = text1.text
num2 = text2.text
Call divide (num1, num2, q, r)
Label1.Caption = "The quotient is " + STR(q) + " with remainder " + STR(r)
End Sub
```

---

## Appendix 3.2.2

### Efficient Programs

There are three aspects of efficiency in programming: 1) hardware utilization (i.e., CPU and memory usage), 2) code authoring, and 3) the user interface.

Measuring hardware utilization is relatively simple and clearly defined. We can measure the number of steps required to complete an operation. The program that requires the fewest steps is the most efficient. The number of steps required may depend on the input. In Appendix 3.2.1, for the non-recursive division algorithm, there are several steps that are required regardless of the size of the divisor or dividend (e.g., the four assignment statements). The number of times the statements within the loop are executed depends on the values for the divisor and dividend. Therefore, measuring the efficiency of this program requires determining the number of steps involved in solving the problem with a set of test data designed to test the range of possibilities. In this case, appropriate test data would include large and small values for dividend and divisor as well as cases in which the dividend and divisor are similar in value and cases in which the divisor is much smaller than the dividend.

A second aspect of efficiency considers the efficient writing of code, which is distinct from the processing issues. In fact, the most efficient program may not be the one that is most efficiently written. For example, consider a program that is to place the numbers from 1 to 100 into an array. You could write this program by writing 100 assignment statements of the form `item(1)=1`, `item(2)=2`, `item(3)=3`, etc. This requires 100 steps in execution. You are more likely, though, to write the program as follows:

```
counter = 1
DO
    item(counter) = counter
    counter = counter + 1
LOOP UNTIL counter > 100
```

From the programmer's perspective, this is a more efficient solution. From the processor perspective, though, this program requires 401 steps in execution and is clearly less efficient! (For each iteration of the loop – 1. assign the value of the counter to the array element, 2. add 1 to the counter, 3. assign the new value of the counter, and 4. compare the counter to the exit value.) This program is also less efficient in memory usage as the value of the counter is stored as well as the required data.

Finally, the third area to consider is the efficiency of the user interface. An interface should not require the user to make repetitive data entry and should avoid the need for unnecessary keystrokes. Programs should avoid situations that require the user to move from keyboard to mouse and input screens should be designed so that data entry is optimized, taking into account a logical flow of data.

---

### Appendix 3.2.3

#### Binary Search Assignment

Plan and write a program to read a list of school clubs or teams, including the name of the club and the name of the teacher/sponsor, from a data file. When the program is executed, the stored data is loaded into a two-dimensional array and sorted in alphabetic order by team name. When the user enters the name of a club, the program uses a recursive binary search to locate and display the name of the teacher/sponsor.

#### Marking Scheme for Binary Search Assignment

Criterion	Mark	Comment
<b>Communication</b>		
Program header contains name, date, and title.	/2	
Indentation is used properly throughout.	/2	
Variables are declared appropriately and commented.	/2	
Variable names are appropriate.	/2	
<b>Thinking/Inquiry</b>		
Problem is fully defined.	/2	
Input and output are fully defined.	/2	
Sub-program parameters are identified.	/2	
Recursive sequence is identified.	/2	
Base case is identified.	/2	
<b>Application</b>		
Data is correctly read from and stored in 2-D array.	/4	
Data is sorted correctly by team name.	/4	
User input is accepted.	/2	
Binary search method returns correct value.	/2	
Binary search method implements recursion correctly.	/4	
The program implements modularity.	/4	

**Note:** Zero value – criterion not attempted. Full Marks – criterion fully implemented. Partial marks according to teacher’s professional judgement of degree of implementation.

### Appendix 3.3.1

#### Approaching a Complex Problem

Top-down programming is really top-down problem solving. It is just as effective when you are trying to solve a problem alone as when you are trying to share a solution with one or several other programmers. It is a way of looking at the problem and searching for a solution. Top-down problem solving moves from the general to the particular.

The essence of top-down programming is to take a difficult problem and simplify it to make it easier to solve. Don’t just sit in front of the keyboard and try bits of code until something starts to work. Think first. Break the overall problem into manageable segments (blocks). Plan a logical sequence of blocks and identify parts that demand more in-depth planning. When you start to code, think of all those modules as procedures. Functions occur when you find a single calculation or evaluation that must be performed repeatedly. Begin with the easy procedures, assuming that the others will work out eventually. It’s a good idea to leave the graphics and screen layout until you have the problem more or less solved. Text output is just fine until you have the problem solved.

---

## Appendix 3.3.2

### The Bumble Bee and the Honeycomb

For this challenge problem, we need to start with a plan – an algorithm. Whatever planning tool you use, the first step is not to decide the number of stripes to place on your bumblebee!

A very busy bee is crawling through a honeycomb, which consists of hexagonal cells (six equal sides) that together form a figure that is also a hexagon (i.e., the lines joining the centres of the outermost cells form a regular hexagon).

Starting at a given cell (randomly selected), the bee wanders randomly from cell to cell throughout the honeycomb. It is trying to visit every cell of the honeycomb but becomes confused and can visit a cell more than once. In one step, it moves from its present cell to any adjacent cell with equal probability. Fortunately for the bee, there are walls around the outside of the honeycomb so that it cannot fall off the outer edge.

Trace the bee's movements from the time that it starts moving until it either visits every cell or it becomes exhausted and goes to sleep.

Conclude the program by showing: the total number of steps taken by the bee, the reason it stopped (exhaustion or it visited every cell) and a representation of the honeycomb showing the number of times that each cell was visited.

#### Strategies

In analysing this problem, students should come up with the following details, and possibly others.

- If the bee is in a cell at the perimeter, there could be adjacent cells on three or four sides, depending on whether the cell is at a corner or along a side.
- If the bee is in an interior cell, there are six adjacent cells.
- We need to find a way to identify all the cells and keep a record of the cells that have been visited.
- We need to keep track of the number of moves that the bee takes.
- We need to determine the number of moves available before the bee become too tired to move.

After brainstorming, students write an algorithm for the overall solution to the problem, as well as either pseudocode or a flowchart for one part of the solution. For example, if the bee is in a cell at the perimeter, there are fewer moves. Depending on which side the cell is on, the directions of the moves are different.

---

### Appendix 3.3.3

#### Challenge Puzzle: Peril At Sea

Your screen represents an area at sea. Using the x-/y-co-ordinate as locations, there are 12 ships randomly scattered across the sea. You are given the speed – in nautical ‘pixelmeters’ per second – which each ship can achieve. One of the ships urgently needs a doctor. Several of the other ships have a doctor on board. There are five pieces of data for each ship:

first:	1 or 0 (1 needs a doctor);
second:	1 or 0 (1 has a doctor);
third:	X-co-ordinate;
fourth:	Y-co-ordinate;
fifth:	speed {in nautical pixelmeters per second}

You are given the first, second, and fifth pieces of data. Let the computer find the third and the fourth. Make sure no two ships are on top of each other. They aren't subs! Find the ship with a doctor that can reach the ship with the patient in the shortest time. Find the co-ordinates where the two ships will meet if they immediately move directly towards each other. Try some graphics to show your various ships and the two ships moving to meet.

There are several jobs in this problem, such as creating the grid to represent the area at sea, creating the graphics of the ships, moving the ships on the grid, and calculating the correct answers based on the information. How do you think we should approach this problem?

#### Strategies

In analysing the problem, students should come up with the following details, and possibly others.

- How big are the ship graphics? The x-/y-coordinates are only part of the each graphic. Should we use a function to determine if each position is permissible?
- What is the distance between any two points on the screen? Would a function be appropriate for determining this distance?
- Should we include speed in the distance calculation or do it separately after determining distance? Would it be more efficient to calculate speed starting with the nearest ship?

After brainstorming, students write an algorithm for the overall solution to the problem. As there are at least two places for functions in this problem, some students write pseudocode solutions for one function while the rest write pseudocode solutions for another. Students should then explain their solutions to other students. This is an opportunity for using communication as well as problem solving skills.

---

## Appendix 3.3.4

### A Recursive Classic – The Towers of Hanoi

The Towers of Hanoi problem is said to be based on a legend in which the monks in a monastery near Hanoi, Vietnam came into possession of 64 golden disks mounted on one of three diamond needles. The disks are in order with the smallest on top and the largest on the bottom. The task given to the monks was to try to move all the disks from one needle to another needle using the following rules.

1. Only one disk may be moved at any one time.
2. No disk may be placed on top of a smaller disk.

According to the legend, after all the disks have been moved, the universe will come to an end. If we suppose that the monks began their task 3000 years ago, and that they have been moving disks constantly at a rate of one per second ever since, calculate the approximate number of years from now to the time when the universe will supposedly end. But don't worry about it too much.

#### Tracing the Development of a Complex Recursive Solution

Think of the needles as needle 1, needle 2, and needle 3. The disks are numbers 1, 2, 3 ... n; 1 is the smallest and n is the largest. Move all the disks from needle 1 to needle 3, using needle 2 to help the process.

If there is only one disk on needle 1 then the problem is insignificant, simply move that disk to needle 3.

Think of a tower with two disks as a one-disk tower placed on top of a larger disk. The instructions for moving such a tower would be:

Move the one-disk tower from needle 1 to needle 2 ( $1 \rightarrow 2$ ).

Move the bottom disk from needle 1 to needle 3 ( $1 \rightarrow 3$ ).

Move the one-disk tower from needle 2 to needle 3 ( $2 \rightarrow 3$ ).

Similarly, think of a tower with three disks as a two-disk tower placed on top of the largest disk. The instructions for moving such a tower would be:

Move the two-disk tower from needle 1 to needle 2; adapt the previous solution ( $1 \rightarrow 3$ ) ( $1 \rightarrow 2$ ) ( $3 \rightarrow 2$ ).

Move the bottom disk from needle 1 to needle 3 ( $1 \rightarrow 3$ ).

Move the two-disk tower from needle 2 to needle 3; once again, adapt the previous solution ( $2 \rightarrow 1$ ) ( $2 \rightarrow 3$ ) ( $1 \rightarrow 3$ ).

The pattern should now be clear. If there are n disks on needle 1, then think of it as a tower of n-1 placed on top of the largest disk. Before you can move disk n to needle 3, you must move disks 1 through n-1 out of the way by moving them to needle 2. Then, after you move disk n to needle 3, you must move disk 1 through n-1 from needle 2 to needle 3.

Break this pattern down into three sub-problems:

1. Move n-1 disks from needle 1 to needle 2.
2. Move n disk from needle 1 to needle 3.
3. Move n-1 disks from needle 2 to needle 3.

Sub-problem 2 is very simple because only one disk is moving. Sub-problems 1 and 3 appear to be very complex but in fact can be divided into further sub-problems until you reach a point where you are only moving one disk. But you will be doing that many times!

---

### Appendix 3.3.4 (Continued)

Examine this recursive algorithm for solving The Towers of Hanoi problem:

If there is only one disk

then

    simply move it from the first needle to the last needle

else

    move n-1 disks from the first needle to the intermediate needle

    move the bottom disk to the third needle

    move the n-1 disks from the intermediate needle to the last needle

Using that algorithm, this is pseudocode for a method called moveTower which has three parameters:

- the height of the tower to be moved;
- the number of the needle on which the tower starts;
- the number of the needle on which the tower finishes.

procedure moveTower (height, start, finish : integer)

variable intermediate : integer // represents the intermediate needle number

if height is equal to 1

then

    just move it from start to finish // exit from the method

else

- determine the location for the intermediate needle by using the fact that the sum of the values of the needle numbers is always six.

    intermediate is assigned the value of  $6 - (\text{start} + \text{finish})$

        move all but one disk from start to intermediate using moveTower method

            moveTower (height - 1, start, intermediate )

        move the bottom disk

            output start , “→”, finish

        move the remaining disks from intermediate to finish using moveTower method

            moveTower (height-1, intermediate, finish )

end procedure

### Strategies

Students should trace the algorithm and the pseudocode together. Understanding them can be difficult.

By tracing the steps, students should be able to determine the number of moves for 1, 2, 3, 4, 5, and 6 disks. After, they should be able to develop a pattern and predict mathematically the number of moves for larger numbers.

Students should also look for problems in the pseudocode, such as what happens when invalid data is entered. What if you entered 0 disks or tried to use four needles? Consideration should be given to developing an efficient solution.

---

## Appendix 3.3.5

### Towers of Hanoi Java Code Sample

```
// The "TowersOfHanoi" class.
/*This program shows a recursive method used to solve the Towers of Hanoi problem.
It demonstrates the moving of a disk as a print statement.
It uses a global variable to record the number of moves needed*/
import java.awt.*;
import hsa.Console;

public class TowersOfHanoi
{
    static Console c; // The output console
    static int count; // global variable for keeping track of the number of moves
    public static void main (String [] args)
    {
        c = new Console ();
        final int DISKS = 3; // change this value to investigate any number of disks
        count = 1; // initialize the count to 1
        moveTower (DISKS, 1, 3);
    } // main method

    public static void moveTower (int height, int start, int finish)
    {
        if (height == 1)
        { // the is final task before exiting from the method
            // move the disk from start to finish
            c.println (count + ": " + start + " --> " + finish);
            // increase the count each time there is output to record a move
            count++;
        }
        else
        { // this is the recursive part of the method
            // determine the location of the intermediate needle
            int intermediate = 6 - (start + finish);
            // move all but 1 disk from start to intermediate
            moveTower (height - 1, start, intermediate);
            // move the bottom disk
            c.println (count + ": " + start + " --> " + finish);
            // increase the count each time there is output to record a move
            count++;
            // move the remaining disks from intermediate to finish using recursion
            moveTower (height - 1, intermediate, finish);
        }
    }
} // TowersOfHanoi class
```

---

## Appendix 3.3.6

### Peer Assessment of Group Algorithm Presentation

Group: \_\_\_\_\_ Assessed by: \_\_\_\_\_

**Key Points** – List four key points or steps in the algorithm.

1	
2	
3	
4	

**Group Explanation** – Complete this checklist.

Was the overall structure explained clearly?	Yes	No
Did the explanation deal with all parts of the algorithm?	Yes	No
Did the explanation explain all parts of the algorithm clearly?	Yes	No
Did the explanation deal with the data being used?	Yes	No
Did the explanation deal with the data clearly?	Yes	No
Did all members of the group contribute to the explanation?	Yes	No

**Identifying Strengths of the Algorithm** – List the two best aspects of this algorithm.

1	
2	

**Identifying Problems in the Algorithm** – List at least one area that you think could be a problem and should be considered.

1	
2	

**Teacher Comments** (on the quality of the assessment)

Area of Assessment	Comment
Key Points	
Group Explanation	
Identifying Strengths	
Identifying Problems	

---

## Appendix 3.3.7

### Rubric for Assessment of Problem Design

Achievement Category	Level 1 (50-59%)	Level 2 (60-69%)	Level 3 (70-79%)	Level 4 (80-100%)
Problem Definition (T/I)	- states and addresses few components of the problem	- states and addresses most components of the problem	- clearly states and addresses most components of the problem	- clearly states and addresses all or almost all components of the problem
Evidence of brainstorming (T/I)	- describes solutions that have limited effectiveness	- describes solutions that are somewhat effective	- describes solutions that are considerably effective	- describes solutions that are highly effective
Refinement of solution (T/I)	- develops few components of the solution	- develops some components of the solution	- develops most components of the solution	- fully develops selected solution
Implementation of methodology (C)	- selected methodology illustrates a few components of the solution	- selected methodology illustrates some components of the solution	- selected methodology illustrates most components of the solution	- selected methodology fully illustrates the solution
Solution of problem (A)	- solves few elements of the problem	- solves some elements of the problem	- solves most elements of the problem	- fully solves problem by the chosen solution

**Note:** A student whose achievement is below Level 1 (50%) has not met the expectations for this assignment or activity.

---

## Appendix 3.4.1

### Types of Errors and Run-Time Behaviour

1. Compile-time Errors
  - a. Syntax errors are mostly typographic errors, such as missing semi-colons, misspelled variable names, or mismatched parentheses.
  - b. Semantic errors (e.g., forgetting to declare/initialize variables and illegal mixing of variable types).
2. Run-time Errors: Errors that occur when program is executing (e.g., division by zero or invalid data).
3. Logic Errors: Your program compiles and runs but does not produce the correct output. These are the hardest errors to find and require that the programmer anticipate the possibility of such errors and use defensive programming to avoid them. A process called testing and debugging can detect logic errors. The goal is to have a robust program that can withstand any input.
  - a. Robustness
    - i. A robust program produces meaningful results from any data set, regardless of how improper.
    - ii. The results may not be correct but we should always be able to perform an operation that is useful to the user. This may be an appropriate error message or the resetting of data within allowable limits and resubmission with explanation.
    - iii. Under no circumstance must the program terminate abnormally.
  - b. Input Validation
    - i. Thorough validation is essential for a robust program. Data entry errors include:
      - Data-entry errors by the user (hitting 7 instead of 4);
      - improper ordering of the data items on a line;
      - format entry errors (e.g., date value of 10/16/2002 or 16/10/2002);The GIGO (garbage in-garbage out) principle applies.

## Appendix 3.4.2

### Test Drivers

The quality of the test data is more important than the quantity. Test drivers are subroutines or methods. Their only purpose is to test the program with a variety of test data. All possible cases and scenarios must be incorporated into the test data. The following must be considered when creating test data:

- 1. The Black-box Method:** Test data is chosen according to the specifications of the problem without regard to the internal details of the program. At a minimum, the test data should include:
- i. Easy values. The program should be debugged with basic data to ensure that it works. Far too often programmers select complex data only for testing.
  - ii. Typical, realistic values. Choose data representative of the normal program use with easy values so that results can be checked by hand.
  - iii. Extreme values. Programmers must assume entry beyond the limits, as it is very easy for counters or array indices to be off by one.
  - iv. Illegal values. Programs must take into account illegal values and provide some indication of the likely errors in input, continuing to perform calculations after disregarding the erroneous data.

---

## Appendix 3.4.2 (Continued)

**2. The Glass-box Method:** All components of the program must be thoroughly tested. In this method, the logical structure of the program is examined. Test data are devised that lead to each and every alternative in each conditional case statement and at the termination of each loop. In larger programs, glass-box testing is not practical, but in smaller programs it becomes an excellent debugging tool. In larger programs, each module is tested individually, starting with the most primitive methods which are used by other methods.

## Appendix 3.4.3

### Debugging and Testing Strategies

Testing merely confirms the presence of errors, not the absence.

Debugging is a process of locating and correcting errors in a program. This can be a painfully slow process unless proper guidelines are followed from the outset. Studies indicate that programmers spend over half of their time finding and correcting errors. Programmers must incorporate methods to avoid these bugs and reduce the time needed to correct them.

1. Place “trace” output statements (e.g. breakpoints or variable watches) at strategic places throughout the program as you develop the code. Inserting debug statements upon entering and exiting subroutines or methods, upon entering a loop, or at other strategic places will avoid errors.
2. Write code that is as logical and readable as possible. Proper style is important but it is also critical for ensuring efficient code.
3. Understand the error completely before fixing it. Use a debugger if one is available with your programming interface. Watch carefully as an error is produced to determine the nature of the error and be able to reproduce the error if necessary. Don’t operate on the “it just worked that way” or the “my friend told me to” principle.
4. Create test drivers for both the black-box and the glass-box methods.

## Appendix 3.4.4

### Assignment

Using the methodology developed in Activity 3, students:

1. code a complete solution for the problem;
2. demonstrate proper debugging techniques by implementing output statements at strategic locations;
3. prepare a set of test data to properly test all conditions, using the black box and glass-box scenarios as guidelines;
4. create a test driver subroutine that runs through the test data developed in step 3.

## Appendix 3.4.5

### Programming Assignment Rubric

Achievement Category	Level 1 (50-59%)	Level 2 (60-69%)	Level 3 (70-79%)	Level 4 (80-100%)
Programming Style (C)	- applies few standard programming practices	- applies some standard programming practices	- applies most standard programming practices	- program conforms to standard programming practices
Program Organization (A)	- demonstrates the use of modularity to a limited degree	- demonstrates the use of modularity to some degree	- demonstrates the use of modularity to a considerable degree	- demonstrates the use of modularity to a high degree
Variables (A)	- few variables have appropriate scope	- some variables have appropriate scope	- most variables have appropriate scope	- all or almost all variables have appropriate scope
Solution of problem (A)	- produces correct output for limited test data cases	- produces correct output for some test data cases	- produces correct output for most test data cases	- produces correct output for all or almost all data cases
Processing Efficiency (T/I)	- avoids a limited number of unnecessary or repetitive processing steps	- avoids some unnecessary or repetitive processing steps	avoids most unnecessary or repetitive processing steps	- uses the minimum number of processing steps
Coding Efficiency (T/I)	- applies a non-recursive and non-efficient solution	- applies a non-recursive but efficient solution	- applies an inefficient recursive algorithm	- efficiently applies a recursive algorithm
User Interface Efficiency (T/I)	- interface requires many additional user actions	- interface requires some additional user actions	- interface requires a few additional user actions	- user interface is highly efficient
Debugging (A)	- contains statements to assist with debugging that have limited effectiveness	- contains statements to assist with debugging that are somewhat effective	- contains considerable statements to assist with debugging that are considerably effective	- contains statements to assist with debugging that are highly effective
Test Data (A)	- develops few test cases	- develops some test cases	- develops data to test most cases	- develops a complete suite of test data

**Note:** A student whose achievement is below Level 1 (50%) has not met the expectations for this assignment or activity.

---

## Appendix 3.4.6

### Unit Test

1. The “Decadent Search” is based on the number 10. To locate an item in a large, sorted list containing  $n$  items, the list is first divided into 10 subgroups each containing  $n/10$  items (i.e., a list of 8742 items would be divided into 10 subgroups containing 875 items). The last group may not always contain as many items as the rest – the number of items per group is always rounded up.  
The first item in each subgroup is compared to the search value to locate the subgroup containing the search item. The group containing the search value is divided into 10 subgroups, in the same way that the first groupings were formed.  
The breaking of the list into 10 subgroups and searching for the correct subgroup continues until the subgroup contains less than 10 items. At that point the items in the subgroup are individually compared to the search value until the search item is located.
  - a) Create a flowchart to illustrate the Decadent Search. (4 marks)
  - b) Write pseudocode for a recursive function that implements the Decadent Search. (4 marks)
  - c) Describe the data that properly tests all conditions for the Decadent Search. (3 marks)
  - d) Use the sample data file, cities.dat, which contains the names of a large number of cities sorted by name and the population of the city. Write a program that allows the user to enter the name of a city and then displays the population of the city. Your program must use a recursive Decadent Search. (10 marks)
  - e) Compare the efficiency of the Decadent Search to the Binary Search. (5 marks)
2. Explain the difference between a syntax error and a runtime error and give an example of each. (4 marks)