

**STUDENT:** \_\_\_\_\_

TASK	OBSERVATION/RATING					
Planning and Presentation	4	3	2	1	0	N/A
Analysis – Hardware	4	3	2	1	0	N/A
Analysis – Software	4	3	2	1	0	N/A
Analysis – Report	4	3	2	1	0	N/A
Presenting/Reporting	4	3	2	1	0	N/A

**STANDARD IS 2 IN EACH APPLICABLE TASK**

**Rating Scale**

<b>4</b>	Exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence.
<b>3</b>	Meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively.
<b>2</b>	Meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately.
<b>1</b>	Meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately.
<b>0</b>	Has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

**TASK CHECKLIST** - criteria for intermediate level

*The student :*

**Preparation and Planning**

- sets goals and describes steps to achieve them
- uses personal initiative to formulate questions and find answers
- accesses a range of relevant in-school/community resources
- interprets, organizes and combines information into a logical sequence
- records information accurately with appropriate supporting detail and using correct technical terms
- plans and uses time effectively
- gathers and responds to feedback regarding approach to task and project status

**ANALYSIS - HARDWARE**

**Content:** analyzes and compares

- two different computer systems (internal components, peripheral devices) based on:
  - client needs
  - information base
  - implementation timelines
  - financial costs
  - workstation requirements
  - inservice training
  - support services
  - warranties

**ANALYSIS - SOFTWARE**

**Content:** analyzes and compares

- three task-specific software packages on the basis of:
  - hardware/operating system requirements
  - user friendliness
  - training/learning effectiveness
  - instructional support
  - command/function parameters screen/page characteristics

- intended use/audience
- intercompatibility with other software

**ANALYSIS REPORT**

**Content:**

- prepares a report that responds to an identified need to provide or upgrade a computer system. The report will provide recommendations and rationale for a particular hardware/software components (recommendation and reasons) that addresses:
  - client needs
  - information base
  - implementation timelines
  - financial costs
  - workstation requirements
  - inservice training
  - support services
  - warranties
  - legal restrictions
- Presenting/Reporting**
  - demonstrates effective use of at least two communication media:
    - e.g., Written: spelling, punctuation, grammar, format (formal/informal)*
    - Oral: voice projection, body language, appearance*
    - Visual: techniques, tools, clarity*
  - maintains acceptable grammatical and technical standards through proofreading and editing
  - provides an introduction that describes the purpose and scope of the project
  - communicates ideas into a logical sequence with sufficient supporting detail
  - states a conclusion by synthesizing the information gathered
  - provides a reference list that includes five or more relevant information sources

**STUDENT:** \_\_\_\_\_

Observations of Students	<b>CRITERIA</b>
4 3 2 1 0	<p><i>The student:</i></p> <p><b><u>Uses the Network</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> logs in and out; uses password (if necessary)</li> <li><input type="checkbox"/> demonstrates the ability to access information and programs on a LAN</li> <li><input type="checkbox"/> demonstrates the ability to download or upload files or data on a LAN</li> <li><input type="checkbox"/> organizes information on a LAN (e.g., create directories, name files)</li> </ul>
4 3 2 1 0	<p><b><u>Relates How Networks Work</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> identifies the LAN's purpose/capabilities</li> <li><input type="checkbox"/> researches and compares network topologies</li> <li><input type="checkbox"/> researches installation and sets up hardware and software of a LAN</li> </ul>
4 3 2 1 0	<p><b><u>Installs and Troubleshoots Software and Hardware</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> designs a plan for installation and configuration of a LAN</li> <li><input type="checkbox"/> installs and connects LAN hardware</li> <li><input type="checkbox"/> installs LAN software</li> <li><input type="checkbox"/> establishes users groups and security rights</li> <li><input type="checkbox"/> installs application software</li> <li><input type="checkbox"/> tests system after installation and make changes as necessary</li> <li><input type="checkbox"/> tests system with users for satisfaction</li> <li><input type="checkbox"/> builds a defence against viruses and intentional or unintentional user exploration</li> </ul>
4 3 2 1 0	<p><b><u>Presents a Proposal for Maintaining a LAN</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> provides technical support for a LAN for a period of time</li> <li><input type="checkbox"/> plans and establishes policies and procedures for:                             <ul style="list-style-type: none"> <li>• ethical use of software</li> <li>• network access, security and backup protection</li> <li>• user access, rights, passwords</li> <li>• file/disk management</li> <li>• software and data upgrades</li> </ul> </li> </ul>

**STANDARD IS 2 IN EACH APPLICABLE TASK**

**Rating Scale**

<b>4</b>	Exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence.
<b>3</b>	Meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively.
<b>2</b>	Meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately.
<b>1</b>	Meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately.
<b>0</b>	Has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

**REFLECTIONS/COMMENTS**

**STUDENT:** \_\_\_\_\_

Observations of Student	CRITERIA
<b>4</b> <b>3</b> <b>2</b> <b>1</b> <b>0</b>	<p><i>The student:</i></p> <p><b><u>Planning</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> identifies user’s needs and determines how the software can be used to meet those needs</li> <li><input type="checkbox"/> defines basic information (key tasks, duration)</li> <li><input type="checkbox"/> links tasks where appropriate</li> <li><input type="checkbox"/> demonstrates ability to set milestones and constraints</li> <li><input type="checkbox"/> demonstrates ability to organize tasks into outline—detailing sub-tasks</li> <li><input type="checkbox"/> assigns resources and creates a base calendar</li> <li><input type="checkbox"/> demonstrates ability to view tasks and outline in a sub-task format</li> </ul>
<b>4</b> <b>3</b> <b>2</b> <b>1</b> <b>0</b>	<p><b><u>Monitoring</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> identifies critical issues</li> <li><input type="checkbox"/> uses data to resolve time restrictions and resource constraints</li> <li><input type="checkbox"/> checks if this meets initial needs of user(s)</li> <li><input type="checkbox"/> makes necessary changes or adjustments</li> <li><input type="checkbox"/> edits, retrieves and manipulates information</li> <li><input type="checkbox"/> generates project reports as required</li> </ul>
<b>4</b> <b>3</b> <b>2</b> <b>1</b> <b>0</b>	<p><b><u>Presenting</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the information management tool to others</li> <li><input type="checkbox"/> discusses the capabilities of the tool</li> <li><input type="checkbox"/> communicates the information in a logical sequence</li> <li><input type="checkbox"/> discusses the capabilities and limitations of using the management tool as it relates to the following:                         <ul style="list-style-type: none"> <li>• user needs</li> <li>• ability to solve the problems of the user</li> </ul> </li> </ul>

**STANDARD IS 2 IN EACH APPLICABLE TASK**

**Rating Scale**

<b>4</b>	Exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence.
<b>3</b>	Meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively.
<b>2</b>	Meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately.
<b>1</b>	Meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately.
<b>0</b>	Has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

**REFLECTION/COMMENTS**

**STUDENT:** \_\_\_\_\_

Observation of Student	CRITERIA
4 3 2 1 0	<p><i>The student</i></p> <p><b>Researches</b> (expert systems, virtual reality [AI interfaces] or other identified AI technologies such as natural languages, robotics, exploratory programming)</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> explains what the AI technology is and how it is affecting society now and in the future</li> <li><input type="checkbox"/> describes the jobs/tasks that the technology can perform in industry and personal living</li> <li><input type="checkbox"/> creates a diagram of the technology and its components, label and give a brief description of components</li> <li><input type="checkbox"/> identifies and gives examples of the advantages and disadvantages of using this type of technology to perform various types of tasks</li> <li><input type="checkbox"/> provides examples of when it would be feasible to use the emerging technology over a human or other present technology to perform a task</li> <li><input type="checkbox"/> describes other important criteria related to specific technology</li> <li><input type="checkbox"/> presents the research in an organized format of choice</li> </ul>
4 3 2 1 0	<p><b>Uses AI Software</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> selects and/or identifies software being used</li> <li><input type="checkbox"/> plans and outlines a task <b>or</b> define and outline the problem</li> <li><input type="checkbox"/> describes the uses of the selected software</li> <li><input type="checkbox"/> demonstrates use of selected software to perform task <b>or</b> solve problem</li> <li><input type="checkbox"/> tests the program developed to perform task <b>or</b> solve problem</li> <li><input type="checkbox"/> adjusts and/or modifies program as a result of test</li> </ul>
4 3 2 1 0	<p><b>Presents the Program</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> identifies purpose of program</li> <li><input type="checkbox"/> demonstrates use of program to complete task <b>or</b> solve problem</li> <li><input type="checkbox"/> explains the details of the program (how it works, challenges to overcome, etc.)</li> <li><input type="checkbox"/> evaluates the end results of the program (what it can and cannot do)</li> <li><input type="checkbox"/> evaluates presentation of the program</li> </ul>

**STANDARD IS 2 IN EACH APPLICABLE TASK**

**Rating Scale**

<b>4</b>	Exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence.
<b>3</b>	Meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively.
<b>2</b>	Meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately.
<b>1</b>	Meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately.
<b>0</b>	Has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

**REFLECTIONS/COMMENTS**

This project can be completed individually or as a group.

### **Research**

The **five major areas** of artificial intelligence comprise **expert systems, natural language, robotics, improved human interfaces (e.g., virtual reality) and exploratory programming**. The area that has received the most attention for commercial use is expert systems. Some areas that expert systems are being functional in are information processing, pattern recognition, game-playing computers and applied fields such as medical diagnosis. **This project will centre around expert systems.**

Within the area of expert systems, you will research and develop a portfolio to gain an introductory knowledge of the concepts. In the second part of the project you will learn how to program a piece of software related to expert systems that will solve a defined problem and you will conclude the project by presenting the program.

#### **Your portfolio should begin with the following:**

- *Topic* – describe the nature of your research (select a specific area/field to study, e.g., medical, computer games industry)
- *Resources* – provide a list of available resources you will use
- *Timeline* – a timeline of when activities are expected to be completed
- *Outcomes* – what you expect to achieve by the end of this project

#### **Continue your portfolio by including research consisting of:**

- an explanation of what expert systems are and how this technology is affecting society now and in the future (e.g., ethics)
- a description of the jobs/tasks expert systems can perform in industry and personal living
- a description of the area/field of expert systems being explored and a detailed diagram or explanation of the expert system
- identification and provision of examples of the advantages and disadvantages of using expert systems to perform various types of tasks in your chosen area
- identification and provision of examples of when it would be feasible to use expert systems over a human or other technology in your chosen area
- description of other important criteria related to expert systems

*Note: Within your above research you should cover topics such as fuzzy logic, state space theory: propositional logic, interpreted language, knowledge base (facts and rules) + inference engine (reasoning ability) = expert systems ability to perform conclusions, artificial intelligence; the use of user interfaces (e.g., virtual reality) in expert systems; and explanation facilities (systems ability to justify conclusions) in expert systems.*

**Application of Software**

Using PROLOG, LISP or another artificial intelligence software package, write a program that solves one of the following problems:

- A farmer is at the river and needs to get to the other side. He has with him a fox, a goose and some grain. He can only take one item with him in the boat at a time. If the fox will eat the goose and the goose will eat the grain how will he get all three of his possessions over to the other side of the river without them being damaged?
- Write a program that allows two people to play tic-tac-toe on the computer
- Write a program that solves a problem as defined by you and/or your teacher

**Presentation of Program**

Present a demonstration of the program to your teacher and/or class and discuss the following:

- identify purpose of program
- demonstrate use of program to solve problem
- explain the details of the program (how it works, challenges to overcome, etc.)
- evaluate the end results of the program (what it can and cannot do)
- evaluate presentation of the program

<b>ASSESSMENT CHECKLIST: TELECOMMUNICATION SYSTEMS INFRASTRUCTURE PRESENTATION/REPORT</b>	<b>INF3180-1</b>
---	------------------

STUDENT: \_\_\_\_\_

<b>STANDARD</b>	Students must demonstrate effective research and presentation/report skills using the criteria as noted in the checklists below by comparing TWO telecommunication systems (wired or wireless or combined). Students working <b>at standard</b> will demonstrate competencies as described in rating scale 3. Students working <b>above standard</b> will demonstrate competencies as described in rating scale 4.
-----------------	--

At Standard	Criteria	Telecommunication System No. 1 _____ <i>The student:</i>	Telecommunication System No. 2 _____ <i>The student:</i>
____/3	Preparation and Planning	<input type="checkbox"/> sets goals and describes steps to achieve them <input type="checkbox"/> uses personal initiative to formulate questions and find answers <input type="checkbox"/> accesses a range of relevant information sources and recognizes when additional information is required <input type="checkbox"/> interprets, organizes and combines information in creative and thoughtful ways <input type="checkbox"/> records information accurately, using appropriate technical terms and supporting detail <input type="checkbox"/> plans and uses time effectively, prioritizing tasks on a consistent basis <input type="checkbox"/> assesses and refines approach to task and project status based on feedback and reflection	<input type="checkbox"/> sets goals and describes steps to achieve them <input type="checkbox"/> uses personal initiative to formulate questions and find answers <input type="checkbox"/> accesses a range of relevant information sources and recognizes when additional information is required <input type="checkbox"/> interprets, organizes and combines information in creative and thoughtful ways <input type="checkbox"/> records information accurately, using appropriate technical terms and supporting detail <input type="checkbox"/> plans and uses time effectively, prioritizing tasks on a consistent basis <input type="checkbox"/> assesses and refines approach to task and project status based on feedback and reflection
____/3	Presentation	<input type="checkbox"/> demonstrates effective use of a variety of communication media <input type="checkbox"/> maintains acceptable grammatical and technical standards through proofreading and editing <input type="checkbox"/> provides an introduction that describes the purpose and scope of the project <input type="checkbox"/> communicates thoughts/feelings/ideas clearly to justify or challenge a position <input type="checkbox"/> states a conclusion by analyzing and synthesizing the information gathered <input type="checkbox"/> gives evidence of adequate research through a reference list including seven or more relevant information sources	<input type="checkbox"/> demonstrates effective use of a variety of communication media <input type="checkbox"/> maintains acceptable grammatical and technical standards through proofreading and editing <input type="checkbox"/> provides an introduction that describes the purpose and scope of the project <input type="checkbox"/> communicates thoughts/feelings/ideas clearly to justify or challenge a position <input type="checkbox"/> states a conclusion by analyzing and synthesizing the information gathered <input type="checkbox"/> gives evidence of adequate research through a reference list including seven or more relevant information sources
____/3	Content	The presentation/report includes the following: <ul style="list-style-type: none"> <li>• application/service provided</li> <li>• transmission system used</li> <li>• software used</li> <li>• standards and protocols used</li> <li>• personnel/expertise required</li> </ul>	The presentation/report includes the following: <ul style="list-style-type: none"> <li>• application/service provided</li> <li>• transmission system used</li> <li>• software used</li> <li>• standards and protocols used</li> <li>• personnel/expertise required</li> </ul>
____/3	Analysis	<ul style="list-style-type: none"> <li>• The presentation/report includes a comparison of the two systems and provides projection of which system will become dominant for a particular target audience/user.</li> </ul>	

<b>Rating Scale</b>	<b>4</b> - Demonstrates initiative that exceeds required techniques/skills	<b>3</b> - Consistently demonstrates all designated techniques/skills, rarely needs prompting	<b>2</b> - Demonstrates all designated techniques/skills, occasionally needs prompting	<b>1</b> - Demonstrates most designated techniques/skills, frequently needs prompting	<b>0</b> - Does not demonstrate designated technique/skill
---------------------	--	---	--	---	--

<b>ASSESSMENT CHECKLIST: TELECOMMUNICATION SYSTEMS INFRASTRUCTURE PRESENTATION/REPORT</b>	<b>INF3180-2</b>
---	------------------

STUDENT: \_\_\_\_\_

<b>STANDARD</b>	Students must demonstrate effective research and presentation/report skills using the criteria as noted in the checklists below by comparing TWO telecommunication systems (wired or wireless or combined). Students working <b>at standard</b> will demonstrate competencies as described in rating scale 3. Students working <b>above standard</b> will demonstrate competencies as described in rating scale 4.
-----------------	--

At Standard	Criteria	Telecommunication System No. 1 _____ <i>The student:</i>	Telecommunication System No. 2 _____ <i>The student:</i>
___/3	Preparation and Planning	<input type="checkbox"/> sets goals and describes steps to achieve them <input type="checkbox"/> uses personal initiative to formulate questions and find answers <input type="checkbox"/> accesses a range of relevant information sources and recognizes when additional information is required <input type="checkbox"/> interprets, organizes and combines information in creative and thoughtful ways <input type="checkbox"/> records information accurately, using appropriate technical terms and supporting detail <input type="checkbox"/> plans and uses time effectively, prioritizing tasks on a consistent basis <input type="checkbox"/> assesses and refines approach to task and project status based on feedback and reflection	<input type="checkbox"/> sets goals and describes steps to achieve them <input type="checkbox"/> uses personal initiative to formulate questions and find answers <input type="checkbox"/> accesses a range of relevant information sources and recognizes when additional information is required <input type="checkbox"/> interprets, organizes and combines information in creative and thoughtful ways <input type="checkbox"/> records information accurately, using appropriate technical terms and supporting detail <input type="checkbox"/> plans and uses time effectively, prioritizing tasks on a consistent basis <input type="checkbox"/> assesses and refines approach to task and project status based on feedback and reflection
___/3	Presentation	<input type="checkbox"/> demonstrates effective use of a variety of communication media <input type="checkbox"/> maintains acceptable grammatical and technical standards through proofreading and editing <input type="checkbox"/> provides an introduction that describes the purpose and scope of the project <input type="checkbox"/> communicates thoughts/feelings/ideas clearly to justify or challenge a position <input type="checkbox"/> states a conclusion by analyzing and synthesizing the information gathered <input type="checkbox"/> gives evidence of adequate research through a reference list including seven or more relevant information sources	<input type="checkbox"/> demonstrates effective use of a variety of communication media <input type="checkbox"/> maintains acceptable grammatical and technical standards through proofreading and editing <input type="checkbox"/> provides an introduction that describes the purpose and scope of the project <input type="checkbox"/> communicates thoughts/feelings/ideas clearly to justify or challenge a position <input type="checkbox"/> states a conclusion by analyzing and synthesizing the information gathered <input type="checkbox"/> gives evidence of adequate research through a reference list including seven or more relevant information sources
___/3	Content	The presentation/report includes the following: <ul style="list-style-type: none"> <li>• target audience</li> <li>• benefits and impacts (individual and societal)</li> <li>• merging and connecting technologies</li> </ul>	The presentation/report includes the following: <ul style="list-style-type: none"> <li>• target audience</li> <li>• benefits and impacts (individual and societal)</li> <li>• merging and connecting technologies</li> </ul>
___/3	Analysis	<ul style="list-style-type: none"> <li>• The presentation/report includes a comparison of the two systems and provides projection of which system will become dominant for a particular target audience/user.</li> </ul>	

<b>Rating Scale</b>	<b>4</b> - Demonstrates initiative that exceeds required techniques/skills	<b>3</b> - Consistently demonstrates all designated techniques/skills, rarely needs prompting	<b>2</b> - Demonstrates all designated techniques/skills, occasionally needs prompting	<b>1</b> - Demonstrates most designated techniques/skills, frequently needs prompting	<b>0</b> - Does not demonstrate designated technique/skill
---------------------	--	---	--	---	--

Student: \_\_\_\_\_

Teacher: \_\_\_\_\_

Module: \_\_\_\_\_

Date: \_\_\_\_\_

CRITERIA	OBSERVATION/ RATING					STANDARD
Management	4	3	2	1	0	3
Teamwork	4	3	2	1	0 NA	3
Content	4	3	2	1	0	3
Equipment and Materials	4	3	2	1	0	3

**STANDARD IS 3 IN EACH APPLICABLE CRITERIA**

**Rating Scale**

*The student:*

- 4 exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence.
- 3 meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively.
- 2 meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately.
- 1 meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately.
- 0 has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

<p><b>CRITERIA</b></p> <p><i>The student:</i></p> <p><b>Management</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> prepares self for task</li> <li><input type="checkbox"/> organizes and works in an orderly manner</li> <li><input type="checkbox"/> interprets and carries out instructions accurately</li> <li><input type="checkbox"/> plans and uses time effectively</li> <li><input type="checkbox"/> adheres to routine procedures</li> </ul> <p><b>Teamwork</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> cooperates with group members</li> <li><input type="checkbox"/> shares work appropriately among group members</li> <li><input type="checkbox"/> negotiates solutions to problems</li> <li><input type="checkbox"/> exhibits basic teamwork skills (e.g., appropriate conduct, leadership, commitment, negotiation, sharing)</li> </ul>	<p><b>Content</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> describes problem being addressed and identifies target audience</li> <li><input type="checkbox"/> lists improvements/benefits of new design</li> <li><input type="checkbox"/> provides appropriate drawings to accurately illustrate:                             <ul style="list-style-type: none"> <li><input type="checkbox"/> component parts</li> <li><input type="checkbox"/> flow of communication</li> <li><input type="checkbox"/> changes/innovation to original (prior) systems</li> </ul> </li> <li><input type="checkbox"/> outline projected impacts on target audience</li> <li><input type="checkbox"/> prepare prototype of design</li> <li><input type="checkbox"/> outline projected costs</li> </ul> <p><b>Equipment and Materials</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> selects and uses appropriate equipment/materials</li> <li><input type="checkbox"/> models safe procedures/techniques</li> <li><input type="checkbox"/> minimizes waste of materials</li> <li><input type="checkbox"/> advises of potential hazards and necessary repairs</li> </ul>
--	---

<p><b>COMMENTS</b></p>
------------------------

STUDENT: \_\_\_\_\_

**STANDARD IS 3 IN EACH APPLICABLE TASK**

**Rating Scale**

*The student:*

<b>4</b>	Exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence.
<b>3</b>	Meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively.
<b>2</b>	Meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately.
<b>1</b>	Meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately.
<b>0</b>	Has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

**REFLECTIONS/COMMENTS:**

Observation of student	CRITERIA
	<b><i>The student:</i></b>
<b>4</b>	<b><u>Researching/Designing/Creating a Web Site</u></b>
<b>3</b>	<input type="checkbox"/> outlines an idea for a web site that is of interest and is appropriate to the audience
<b>2</b>	<input type="checkbox"/> finds and collects information on the topics of interest
<b>1</b>	<input type="checkbox"/> identifies effective elements to be used in web site
<b>0</b>	<input type="checkbox"/> organizes data for linking to other web pages
<b>4</b>	<input type="checkbox"/> constructs a web site that is visually pleasing, incorporating:
<b>3</b>	– text, graphics, links, anchors
<b>2</b>	– a functioning advanced feature; e.g., sound, animation
<b>1</b>	– a suitable layout for intended purpose
<b>0</b>	– accepted guidelines such as:
	• attractive, yet simple
	• user friendly
	• feedback option
	<input type="checkbox"/> formats information in an acceptable and/or creative style
	<input type="checkbox"/> tests and debugs web site
<b>4</b>	<b><u>Presenting/Documenting Advanced Feature(s)</u></b>
<b>3</b>	<input type="checkbox"/> presents information on how to implement the advanced feature in an understandable manner
<b>2</b>	<input type="checkbox"/> presents web site to others
<b>1</b>	<input type="checkbox"/> assists others to duplicate the special feature(s)
<b>0</b>	<input type="checkbox"/> properly cites all resources
<b>4</b>	<b><u>Maintaining/Enhancing a Web Site</u></b>
<b>3</b>	<input type="checkbox"/> evaluates the impact of the web site
<b>2</b>	<input type="checkbox"/> identifies which areas of web sites need monitoring
<b>1</b>	<input type="checkbox"/> updates web site
<b>0</b>	<input type="checkbox"/> edits web site (text, graphics, etc.)
	<input type="checkbox"/> enhances web site by improving or adding special feature(s)

STUDENT: \_\_\_\_\_

**STANDARD IS 3 IN EACH APPLICABLE TASK**

**Rating Scale**

*The student:*

<b>4</b>	Exceeds defined outcomes. Plans and solves problems effectively and creatively in a self-directed manner. Tools, materials and/or processes are selected and used efficiently, effectively and with confidence. <i>Leads others to contribute to team goals.</i>
<b>3</b>	Meets defined outcomes. Plans and solves problems in a self-directed manner. Tools, materials and/or processes are selected and used efficiently and effectively. <i>Works cooperatively and contributes ideas and suggestions that enhance team effort.</i>
<b>2</b>	Meets defined outcomes. Plans and solves problems with limited assistance. Tools, materials and/or processes are selected and used appropriately. <i>Works cooperatively to achieve team goals.</i>
<b>1</b>	Meets defined outcomes. Follows a guided plan of action. A limited range of tools, materials and/or processes are used appropriately. <i>Works cooperatively.</i>
<b>0</b>	Has not completed defined outcomes. Tools, materials and/or processes are used inappropriately.

**REFLECTIONS/COMMENTS:**

Observation of student	CRITERIA
	<b><i>The student:</i></b>
4 3 2 1 0	<p><b><u>Accessing Information</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> finds and uses existing services of interest; e.g., e-mail</li> <li><input type="checkbox"/> follows proper netiquette procedure</li> <li><input type="checkbox"/> reports back findings</li> </ul>
4 3 2 1 0	<p><b><u>Designing/Creating</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> designs a functional communication system</li> <li><input type="checkbox"/> builds a functional communication system</li> <li><input type="checkbox"/> maintains the system for an agreed period of time</li> <li><input type="checkbox"/> presents information on how to implement the communication system in an understandable manner</li> <li><input type="checkbox"/> presents system details to others</li> </ul>
4 3 2 1 0	<p><b><u>Communication System as an Operator/Manager</u></b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> configures hardware and software</li> <li><input type="checkbox"/> maintains files and user accounts</li> <li><input type="checkbox"/> troubleshoots and diagnoses problems</li> <li><input type="checkbox"/> offers user/client support service</li> <li><input type="checkbox"/> monitors/updates information and messages</li> </ul>

**ASSESSMENT CHECKLIST: COMPUTER SCIENCE 1**

**INF1210-1**

STUDENT: \_\_\_\_\_

<b>STANDARD:</b>	Students working <b>at standard</b> must demonstrate a general understanding of the nature, approaches, areas of interest and algorithmic basis of the discipline of computer science. As the main focus of computer science is the utilization of algorithmic approaches to problem solving, most of the emphasis of this course is on the ability to understand design, develop, implement and test algorithmic solutions to problems amenable to structured programming approaches. As this course is designed to be taught in conjunction with Programming 1 and Programming 2 this assessment checklist dovetails with the checklist for those courses. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
------------------	---

At Standard	<i>Computer Science 1</i>	
<b>2</b>	<p><b><u>Background Knowledge and Skills:</u></b> The student:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an introductory understanding of the nature, approaches and areas of interest of computer science</li> <li><input type="checkbox"/> demonstrates an understanding of the nature of the algorithm and the ability to use IPO analysis and design techniques to construct algorithms and programs that reflect the structured programming paradigm</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to illustrate how computer scientists use computer languages to convert an algorithm into a form that can be executed by a computer</li> <li><input type="checkbox"/> demonstrates through the use of block diagrams, or other appropriate techniques, how a computer uses a program to accept data from the input section, use the CPU and memory to process the data and the output section to display the processed data</li> </ul>
<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Design/Development Stage:</u></b> The student designs an algorithmic solution to a problem that:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to analyze a problem and identify the initial state, the final state and the input, output and processing requirements of the problem</li> <li><input type="checkbox"/> identifies the appropriate literals, constants and variables needed to accommodate the input, output and processing data to be handled in the problem solution</li> <li><input type="checkbox"/> identifies any idioms required by the program</li> <li><input type="checkbox"/> identifies the structured programming constructs or control structures needed by the program (sequential, selection and repetitive constructs)</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to design a user interface that prompts for and accepts input, and displays the output in an appropriate format</li> <li><input type="checkbox"/> creates an algorithm, using an appropriate form (flowchart, psuedo-code, IPO chart) that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> passes the “fail on paper” test</li> </ul>
<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Implementation Phase:</u></b> The student converts an algorithm into a program that requires the following:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization</b> <ul style="list-style-type: none"> <li>• introduces program to user giving purpose and process</li> <li>• introduces user to interface</li> <li>• declares and initializes constants and variables identified in the algorithm</li> </ul> </li> <li><input type="checkbox"/> <b>Input</b> <ul style="list-style-type: none"> <li>• prompts user for input</li> <li>• accepts input</li> <li>• uses appropriate decision and iterative constructs to check input (type and range)</li> </ul> </li> <li><input type="checkbox"/> <b>Processing</b> <ul style="list-style-type: none"> <li>• updates variables to reflect input</li> <li>• carries out calculations, comparisons, incrementing required by the updating</li> <li>• uses appropriate sequential, selection and repetitive constructs to process data</li> <li>• updates information to be displayed to user</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Output</b> <ul style="list-style-type: none"> <li>• displays updated information to user in appropriate format</li> <li>• highlights significant information</li> </ul> </li> <li><input type="checkbox"/> <b>Linking/Termination</b> <ul style="list-style-type: none"> <li>• determines if a final state has been achieved</li> <li>• if a final state has not been achieved, loop back to input to repeat the input/processing/output cycle</li> <li>• if a final state has been achieved, terminate the program in an appropriate manner</li> </ul> </li> </ul>

<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Execution Phase:</u></b> The student tests and completes the documentation of a program created earlier by the student from an algorithm. These processes should do the following:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; vertical-align: top;"> <p><input type="checkbox"/> <b>Testing</b></p> <ul style="list-style-type: none"> <li>• program is tested with data that produces known results</li> <li>• program is tested with boundary data to test for OB1 (Off By 1) errors</li> <li>• program is tested with aberrant data to test error checking</li> <li>• all necessary modifications are made</li> </ul> </td> <td style="width: 50%; vertical-align: top;"> <p><input type="checkbox"/> <b>Documentation</b></p> <ul style="list-style-type: none"> <li>• does internal and external documentation through all stages employing an appropriate reporting approach</li> <li>• presents statement of problem and algorithm to show problem solution</li> <li>• outlines scope or limits of the program solution</li> <li>• presents documented code listings</li> </ul> </td> </tr> </table>	<p><input type="checkbox"/> <b>Testing</b></p> <ul style="list-style-type: none"> <li>• program is tested with data that produces known results</li> <li>• program is tested with boundary data to test for OB1 (Off By 1) errors</li> <li>• program is tested with aberrant data to test error checking</li> <li>• all necessary modifications are made</li> </ul>	<p><input type="checkbox"/> <b>Documentation</b></p> <ul style="list-style-type: none"> <li>• does internal and external documentation through all stages employing an appropriate reporting approach</li> <li>• presents statement of problem and algorithm to show problem solution</li> <li>• outlines scope or limits of the program solution</li> <li>• presents documented code listings</li> </ul>
<p><input type="checkbox"/> <b>Testing</b></p> <ul style="list-style-type: none"> <li>• program is tested with data that produces known results</li> <li>• program is tested with boundary data to test for OB1 (Off By 1) errors</li> <li>• program is tested with aberrant data to test error checking</li> <li>• all necessary modifications are made</li> </ul>	<p><input type="checkbox"/> <b>Documentation</b></p> <ul style="list-style-type: none"> <li>• does internal and external documentation through all stages employing an appropriate reporting approach</li> <li>• presents statement of problem and algorithm to show problem solution</li> <li>• outlines scope or limits of the program solution</li> <li>• presents documented code listings</li> </ul>		

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required knowledge/techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	---	--	---	--	---

## An Ecology Simulation

A biology teacher at your school has approached you to write a simple ecology simulation for an introductory science class. The ecology is to have the following characteristics:

- the setting for the ecology is to be a closed system, such as an island
- the ecology is to have one type of plant life, one type of herbivore and one type of carnivore
- the students (the users) will be allowed to set the size of the island (in hectares), the initial number of plants, the initial number of herbivores and the initial number of carnivores
- the simulation is to run in yearly cycles; during each year, the carnivores are to prey on the herbivores and the herbivores are to feed on the plant life. In addition, each species is to reproduce over the course of the year
- each carnivore requires 50 herbivores a year to survive, each herbivore requires 5000 plants a year to survive and each hectare of land can support up to 100 000 plants
- the carnivores' rate of reproduction is 1 to 2 (i.e., 1 carnivore can produce 2 offspring—assuming that there are a minimum of 2 carnivores in the system), the herbivores' rate of reproduction is 1 to 6 and the plants' rate of reproduction is 1 to 50
- assume that each species only consumes what it needs to survive (i.e., that the carnivores only eat 50 herbivores a year), that plants and animals that cannot get the food they need to survive die, that all calculations are done at the end of each year and that predation occurs prior to reproduction
- run the simulation for 10 years, or until the user wishes to exit or until the ecology “crashes”; (This ecology can be said to have crashed when one or more of the species dies out. Note: Simple ecologies are very unstable and are prone to crashing.)
- display the results of each cycle as a row in a table. As students are not expected to be familiar with arrays and/or vectors, this data will have to be displayed as the simulation is executing with each “year’s” data being displayed immediately after it is calculated.

Design and develop an algorithm that:

- employs problem parsing to analyze the problem
- uses the expanded IPO paradigm to structure the algorithm
- uses the top-down, step-wise refinement approach to add detail to the algorithm
- uses an appropriate nomenclature, such as flowcharting or pseudo-code
- sketches a screen display(s) that illustrates the user interface outlined in the algorithm
- passes a walk-through or failed-on-paper test.

Translate the algorithm into an executable program that:

- maintains the logic and structure of the algorithm
- employs good structured programming practices (structured constructs and blocks)
- incorporates the user interface designed in the analysis and design stages
- is built in executable increments a few statements at a time
- has adequate internal and external documentation.

Test and implement the program by:

- executing the program with data known to produce specific output
- executing the program with aberrant data
- checking for congruency with the original requirements of the problem.

**For assessment standards and criteria, see Assessment Checklist: Computer Science 1, INF1210-1.**

STUDENT: \_\_\_\_\_

<b>STANDARD</b>	Students working <b>at standard</b> must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
-----------------	---

At Standard	<i>Object-oriented Programming 1</i>		
<b>2</b>	<p><b>Problem-solving Phase:</b> The student:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures) and classes required in the program</li> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures) and classes required in the program</li> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures) and classes required in the program</li> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>		
<b>3</b>	<p><b>Implementation Phase:</b> The student creates a minimum of three programs containing the following—see sample assignments for Object-oriented Programming 1:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification of existing classes</li> <li>• construction of new classes</li> <li>• creation of class libraries</li> <li>• instantiation of new objects</li> <li>• data stored in appropriate class data members</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of data stored in arrays</li> </ul> </li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> <li>• data is transferred to objects (object oriented only)</li> <li>• classes are accessed from class libraries</li> <li>• data is updated and transferred to and from objects</li> <li>• class members functions are used for processing</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• data is output from class data members</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user’s guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> <li>• class relationship diagrammed</li> </ul> </li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification of existing classes</li> <li>• construction of new classes</li> <li>• creation of class libraries</li> <li>• instantiation of new objects</li> <li>• data stored in appropriate class data members</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of data stored in arrays</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> <li>• data is transferred to objects (object oriented only)</li> <li>• classes are accessed from class libraries</li> <li>• data is updated and transferred to and from objects</li> <li>• class members functions are used for processing</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• data is output from class data members</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user’s guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> <li>• class relationship diagrammed</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification of existing classes</li> <li>• construction of new classes</li> <li>• creation of class libraries</li> <li>• instantiation of new objects</li> <li>• data stored in appropriate class data members</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of data stored in arrays</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> <li>• data is transferred to objects (object oriented only)</li> <li>• classes are accessed from class libraries</li> <li>• data is updated and transferred to and from objects</li> <li>• class members functions are used for processing</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• data is output from class data members</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user’s guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> <li>• class relationship diagrammed</li> </ul> </li> </ul>		

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	---	--	---	--	---

Your school employs a number of students to run the copy machines. The employees are paid \$5.00 per hour. The tax rate varies according to the amount earned—more than \$200.00 per week is calculated at 42%, more than \$100.00 and less than or equal to \$200.00 is calculated at 30%, and less than or equal to \$100.00 pays no tax. Overtime is paid to employees—time and a half to those working over 40 hours per week.

### ASSIGNMENT A

Design and code a program that uses an instructor-supplied class (EmployeeClass) to store the data and calculate the wages for three employees for a week. The member functions or methods of the class should:

- gather demographic data on each employee—surname, first name and employee number
- store the number of hours worked per week
- calculate the gross pay and deductions, and return both to the main program.

These values should be passed to a subprogram(s) in the client program that prints the hours worked, gross pay, deductions and net pay. The subprogram(s) should include appropriate derived data types for error trapping on data entry. The subprogram(s) should produce the following output:

#### Individual Employee Reports for (your school)

Employee #1	Employee #2	Employee #3
Name: Harry Smith	Name: Gordon Elliot	Name: Ken East
Hours Worked: 40	Hours Worked: 50	Hours Worked: 10
Gross Pay: 200.00	Gross Pay: 275.00	Gross Pay: 50.00
Deductions: 66.67	Deductions: 115.50	Deductions: 0.00
Net Pay: 133.33	Net Pay: 160.50	Net Pay: 50.00

(continued)

**Summaries for the data for each employee:**

<b>Name</b>	<b>Hours Worked</b>	<b>Total Gross</b>	<b>Total Deductions</b>	<b>Net Pay</b>
Harry Smith	40	200.00	66.67	133.33
Gordon Elliot	50	275.00	115.50	160.50
Ken East	10	50.00	0.00	50.00

**Surname first for the data for each employee:**

<b>Name</b>	<b>Hours Worked</b>	<b>Total Gross</b>	<b>Total Deductions</b>	<b>Net Pay</b>
Smith, Harry	40	200.00	66.67	133.33
Elliot, Gordon	50	275.00	115.50	160.50
East, Ken	10	50.00	0.00	50.00

**School Summary:**

<b>Total Gross</b>	<b>Total Deductions</b>	<b>Total Net</b>
525.00	182.17	343.83

**For assessment standards and criteria, see Assessment Checklist: Object-oriented Programming 1, INF2220-1.**

**ASSIGNMENT B**

Modify the above design and program so that the subprogram(s) used to output the Individual Employee Reports is (are) added to the instructor-supplied class as (a) member function(s).

Further modify the program by creating a class that uses data from the EmployeeClass to generate the summaries.

**For assessment standards and criteria, see Assessment Checklist: Object-oriented Programming 1, INF2220-1.**

STUDENT: \_\_\_\_\_

<b>STANDARD:</b>	Students working <b>at standard</b> must demonstrate an adequate intermediate understanding of the nature, approaches, areas of interest and algorithmic basis of the discipline of computer science. As the main focus of computer science is the utilization of algorithmic approaches to problem solving, most of the emphasis of this course is on the ability to understand design, develop, implement and test algorithmic solutions to problems amenable to introductory modular programming approaches. As this course is designed to be taught in conjunction with Programming 3 and Programming 4, this assessment checklist dovetails with the checklists for those courses. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
------------------	--

At Standard	<i>Computer Science 2</i>		
<b>2</b>	<p><b><u>Background Knowledge and Skills:</u></b> The student:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of the qualitative and quantitative trends in the development of computer technology</li> <li><input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding approaches, such as problem decomposition—into subprograms—to the simpler top-down and step-wise refinement problem parsing techniques developed at earlier levels. The student now constructs algorithms and programs that reflect both the structured and the modular programming paradigms</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of how machines employing the von Neumann computer architecture are programmed to process stored data at the machine language level. This is likely best done through the manipulation of a virtual computer with its own “toy” machine language or Assembly language. This virtual computer could either be a computer simulation of a simple computer, a non-computer simulation of a simple computer or a paper-and-pencil depiction of a simple computer</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of the qualitative and quantitative trends in the development of computer technology</li> <li><input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding approaches, such as problem decomposition—into subprograms—to the simpler top-down and step-wise refinement problem parsing techniques developed at earlier levels. The student now constructs algorithms and programs that reflect both the structured and the modular programming paradigms</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of how machines employing the von Neumann computer architecture are programmed to process stored data at the machine language level. This is likely best done through the manipulation of a virtual computer with its own “toy” machine language or Assembly language. This virtual computer could either be a computer simulation of a simple computer, a non-computer simulation of a simple computer or a paper-and-pencil depiction of a simple computer</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of the qualitative and quantitative trends in the development of computer technology</li> <li><input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding approaches, such as problem decomposition—into subprograms—to the simpler top-down and step-wise refinement problem parsing techniques developed at earlier levels. The student now constructs algorithms and programs that reflect both the structured and the modular programming paradigms</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of how machines employing the von Neumann computer architecture are programmed to process stored data at the machine language level. This is likely best done through the manipulation of a virtual computer with its own “toy” machine language or Assembly language. This virtual computer could either be a computer simulation of a simple computer, a non-computer simulation of a simple computer or a paper-and-pencil depiction of a simple computer</li> </ul>		
<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Design/Development Stage:</u></b> The student designs an algorithmic solution to a problem that employs the tactics developed in the Computer Science 1 course and in addition:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to decompose a problem into simpler subproblems and to continue that process until the problem is decomposed to the point where known idioms can be employed</li> <li><input type="checkbox"/> develops a data dictionary that identifies the data types and any derived and abstract data structures needed by the algorithm</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> uses appropriate subprograms—procedures and functions—with a high level of cohesion and relatively low level of coupling</li> <li><input type="checkbox"/> uses an appropriate construction technique, such as structure diagrams, HIPO charting, structured charting or Warnier–Orr diagrams to identify levels of abstraction, scope and coupling considerations</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to decompose a problem into simpler subproblems and to continue that process until the problem is decomposed to the point where known idioms can be employed</li> <li><input type="checkbox"/> develops a data dictionary that identifies the data types and any derived and abstract data structures needed by the algorithm</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> uses appropriate subprograms—procedures and functions—with a high level of cohesion and relatively low level of coupling</li> <li><input type="checkbox"/> uses an appropriate construction technique, such as structure diagrams, HIPO charting, structured charting or Warnier–Orr diagrams to identify levels of abstraction, scope and coupling considerations</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to decompose a problem into simpler subproblems and to continue that process until the problem is decomposed to the point where known idioms can be employed</li> <li><input type="checkbox"/> develops a data dictionary that identifies the data types and any derived and abstract data structures needed by the algorithm</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> uses appropriate subprograms—procedures and functions—with a high level of cohesion and relatively low level of coupling</li> <li><input type="checkbox"/> uses an appropriate construction technique, such as structure diagrams, HIPO charting, structured charting or Warnier–Orr diagrams to identify levels of abstraction, scope and coupling considerations</li> </ul>		
<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Implementation Phase:</u></b> The student converts an algorithm into a program that employs all of the tactics developed in the Computer Science 1 course and in addition should:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> utilize subprograms, such as functions and procedures:                             <ul style="list-style-type: none"> <li>• to create a modular program</li> <li>• to create functionally abstract blocks of code</li> </ul> </li> <li><input type="checkbox"/> utilize appropriate code building approaches, such as:                             <ul style="list-style-type: none"> <li>• stub programming techniques or its equivalent</li> <li>• iterative prototyping</li> <li>• creation of specialized libraries to promote information hiding</li> </ul> </li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> utilize derived or structured data types, such as arrays, vectors, structs and records to:                             <ul style="list-style-type: none"> <li>• input, process and output related, structured or matrix data</li> </ul> </li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> utilize subprograms, such as functions and procedures:                             <ul style="list-style-type: none"> <li>• to create a modular program</li> <li>• to create functionally abstract blocks of code</li> </ul> </li> <li><input type="checkbox"/> utilize appropriate code building approaches, such as:                             <ul style="list-style-type: none"> <li>• stub programming techniques or its equivalent</li> <li>• iterative prototyping</li> <li>• creation of specialized libraries to promote information hiding</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> utilize derived or structured data types, such as arrays, vectors, structs and records to:                             <ul style="list-style-type: none"> <li>• input, process and output related, structured or matrix data</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> utilize subprograms, such as functions and procedures:                             <ul style="list-style-type: none"> <li>• to create a modular program</li> <li>• to create functionally abstract blocks of code</li> </ul> </li> <li><input type="checkbox"/> utilize appropriate code building approaches, such as:                             <ul style="list-style-type: none"> <li>• stub programming techniques or its equivalent</li> <li>• iterative prototyping</li> <li>• creation of specialized libraries to promote information hiding</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> utilize derived or structured data types, such as arrays, vectors, structs and records to:                             <ul style="list-style-type: none"> <li>• input, process and output related, structured or matrix data</li> </ul> </li> </ul>		

<b>2</b>	<p><b>Utilization of Knowledge and Skills: Execution Phase:</b>                  The student tests and completes the documentation of a program created earlier by the student from an algorithm. These processes should employ all of the tactics developed in the Computer Science 1 course and in addition:</p> <table style="width: 100%; border: none;"> <tr> <td style="vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Testing</b> <ul style="list-style-type: none"> <li>• beta tests the program with knowledgeable users (another class member)</li> </ul> </li> <li><input type="checkbox"/> <b>Implementation</b> <ul style="list-style-type: none"> <li>• program is presented to end user with appropriate instructions</li> </ul> </li> <li><input type="checkbox"/> <b>Maintenance</b> <ul style="list-style-type: none"> <li>• user feedback is solicited to guide future revisions</li> </ul> </li> </ul> </td> <td style="vertical-align: top; padding-left: 20px;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Documentation</b> <ul style="list-style-type: none"> <li>• describes problems encountered during design, development, production and testing</li> <li>• describes possible updates based on user feedback</li> </ul> </li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Testing</b> <ul style="list-style-type: none"> <li>• beta tests the program with knowledgeable users (another class member)</li> </ul> </li> <li><input type="checkbox"/> <b>Implementation</b> <ul style="list-style-type: none"> <li>• program is presented to end user with appropriate instructions</li> </ul> </li> <li><input type="checkbox"/> <b>Maintenance</b> <ul style="list-style-type: none"> <li>• user feedback is solicited to guide future revisions</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Documentation</b> <ul style="list-style-type: none"> <li>• describes problems encountered during design, development, production and testing</li> <li>• describes possible updates based on user feedback</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Testing</b> <ul style="list-style-type: none"> <li>• beta tests the program with knowledgeable users (another class member)</li> </ul> </li> <li><input type="checkbox"/> <b>Implementation</b> <ul style="list-style-type: none"> <li>• program is presented to end user with appropriate instructions</li> </ul> </li> <li><input type="checkbox"/> <b>Maintenance</b> <ul style="list-style-type: none"> <li>• user feedback is solicited to guide future revisions</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Documentation</b> <ul style="list-style-type: none"> <li>• describes problems encountered during design, development, production and testing</li> <li>• describes possible updates based on user feedback</li> </ul> </li> </ul>		

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required knowledge/ techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	--	--	---	--	---

**An Ecology Simulation (An enhanced version of the sample assignment outlined in INF\_CSSAMP1)**

A biology teacher at your school has approached you to write a simple ecology simulation for an introductory science class. The ecology is to have the following characteristics:

- the setting for the ecology is to be a closed system, such as an island
- the ecology is to have one type of plant life, one type of herbivore and one type of carnivore
- the students (the users) will be allowed to set the size of the island (in hectares), the initial number of plants, the initial number of herbivores and the initial number of carnivores
- the simulation is to run in yearly cycles; during each year, the carnivores are to prey on the herbivores and the herbivores are to feed on the plant life. In addition, each species is to reproduce over the course of the year. The processes of predation and reproduction are ideally suited to be handled as functional abstractions through the use of subproblems and subprograms
- each carnivore requires 50 herbivores a year to survive, each herbivore requires 5000 plants a year to survive and each hectare of land can support up to 100 000 plants
- the carnivores' rate of reproduction is 1 to 2 (i.e., 1 carnivore can produce 2 offspring—assuming that there are a minimum of 2 carnivores in the system), the herbivores' rate of reproduction is 1 to 6 and the plants' rate of reproduction is 1 to 50
- assume that each species only consumes what it needs to survive (i.e., that the carnivores only eat 50 herbivores a year), that plants and animals that can not get the food they need to survive die, that all calculations are done at the end of each year and that predation occurs prior to reproduction
- run the simulation for 10 years, or until the user wishes to exit or until the ecology “crashes”; (This ecology can be said to have crashed when one or more of the species dies out. Note: Simple ecologies are very unstable and are prone to crashing.)
- display the results of each cycle as a row in a table. As students are expected to be familiar with arrays and/or vectors, this data should be stored in an appropriate derived data type, such as an array, vector, record or struc and recalled when needed
- display the results as either a line graph or bar chart. The data for this display would have to be extracted from a derived data type, such as an array.

Design and develop an algorithm that:

- employs problem parsing to analyze the problem and decompose it into a hierarchy of subproblems
- identifies similar subproblems that might be amenable to reusable code
- uses the expanded IPO paradigm to structure the first level of the algorithm
- uses a top-down, step-wise refinement approach to add detail to each level of subproblem in the algorithm
- uses an appropriate nomenclature, such as HIPO or structure charts, Warnier–Orr diagrams, or pseudo-code to outline the algorithm
- incorporates an appropriate data dictionary and outlines the required data structures
- sketches a screen display(s) that illustrates the user interface and the format of the graphical output
- passes a walk-through or failed-on-paper test.

Translate the algorithm into an executable program that:

- maintains the logic and structure of the algorithm
- employs good modular programming practices (loosely coupled, highly cohesive subprograms with extensive use of local data)
- incorporates the user interface and displays designed in the analysis and design stages
- uses a technique, such as stub programming to translate the various levels of the algorithm into executable code:
  - uses top–down coding to build a program in executable increments a few statements at a time
  - uses bottom–up coding to build subprograms for testing
  - makes appropriate use of derived data types to input, process and output information
- has adequate internal and external documentation.

Test and implement the program by:

- executing the program with data known to produce specific output
- executing the program with aberrant data
- checking for congruency with the original requirements of the problem
- beta testing the program with knowledgeable users, such as another class member
- providing adequate end user instructions
- soliciting user feedback to guide future revisions.

**For assessment standards and criteria, see Assessment Checklist: Computer Science 2, INF2210–1.**

**ASSESSMENT CHECKLIST: OBJECT-ORIENTED PROGRAMMING 2**

**INF3220-1**

STUDENT: \_\_\_\_\_

<b>STANDARD</b>	Students working <b>at standard</b> must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
-----------------	---

At Standard	<i>Object-oriented Programming 2</i>	
<b>2</b>	<p><b>Problem-solving Phase:</b> The student:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>
<b>3</b>	<p><b>Implementation Phase:</b> The student creates a minimum of three programs containing the following—see sample assignment for Object-oriented Programming 2:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• character, string, integer, real, Boolean and user-defined variables</li> <li>• numeric, character, string and parameter constants</li> <li>• data entered through keyboard entry and assignment statements</li> <li>• appropriate use of scope—local and global variables</li> <li>• data is stored in appropriate derived data types and class data members</li> <li>• error trapping occurs, using appropriate approaches, including derived data types and class member functions</li> <li>• modification of existing classes</li> <li>• construction of new classes</li> <li>• creation of class libraries</li> <li>• instantiation of new objects</li> <li>• data stored in appropriate class data members</li> <li>• construction of templated classes</li> <li>• appropriate use of composition or containment</li> <li>• construction of base and derived classes</li> <li>• use of constructor and operator overloading</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division, absolute numbers, exponentiation</li> <li>• truncates or rounds to a prescribed number of decimal places</li> <li>• decision-making constructs</li> <li>• iterative and recursive constructs</li> <li>• predetermine, pre-check and post-check looping constructs</li> </ul> </li> </ul>	

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	---	--	---	--	---

Your school employs a number of students to run the copy machines. The employees are paid a base rate of \$5.00 per hour. The tax rate varies according to the amount earned—more than \$200.00 per week is calculated at 42%, more than \$100.00 and less than or equal to \$200.00 is calculated at 30%, and less than or equal to \$100.00 pays no tax. Overtime is paid to employees—time and a half to those working over 40 hours per week.

Your school’s duplicating department has grown to the point where it is taking in work from other schools. To accommodate this extra work, two students have been promoted to employee/manager. Their base rate of pay is \$8.00 per hour. In addition, they get a 10% increase on their base rate of pay for every hour (or part hour) that they supervise three or more students.

Design and code a program that uses a class (EmployeeClass) to store the data and to calculate wages for three regular employees for a week. The member functions or methods of the class should:

- gather demographic data on each employee—surname, first name and employee number
- store the number of hours worked per week over the course of the month—four weeks equals one month
- calculate the gross pay and deductions, and return both to the main program.

These values should be passed to a subprogram(s) in the client program that prints the hours worked, gross pay, deductions and net pay. The subprogram(s) should include appropriate derived data types for error trapping on data entry. The subprogram(s) should produce the following output:

**Individual Employee Reports for (your school)**

Employee #1	Employee #2	Employee #3
Name: Harry Smith	Name: Gordon Elliot	Name: Ken East
Hours Worked: 40	Hours Worked: 50	Hours Worked: 10
Gross Pay: 200.00	Gross Pay: 275.00	Gross Pay: 50.00
Deductions: 66.67	Deductions: 115.50	Deductions: 0.00
Net Pay: 133.33	Net Pay: 160.50	Net Pay: 50.00

(continued)

Construct a derived class based on EmployeeClass to store the data and to calculate wages for the two employee/managers for a month. The member functions or methods of this derived class should:

- gather demographic data on each employee/manager—surname, first name and employee/manager number
- store the number of hours worked per week
- store the number of hours the employee/manager supervised three or more employees
- calculate the regular pay, supervisor pay and deductions; and return these values to the client program.

These values should be passed to a subprogram(s) in the client program that prints the hours worked, the regular and supervisory pay, the deductions and the net pay. The subprogram(s) should include appropriate derived data types for error trapping on data entry. The subprogram(s) should produce the following output:

#### Individual Employee/Manager Reports for (your school)

##### Employee/Manager #1

Name: Sam Handwich  
 Regular Hours: 30  
 Supervisory Hours: 5  
 Regular Pay: 240.00  
 Supervisory Pay: 44.00  
 Deductions: 119.28  
 Net Pay: 164.72

##### Employee/Manager #2

Name: Tam Jart  
 Regular Hours: 25  
 Supervisory Hours: 10  
 Regular Pay: 200.00  
 Supervisory Pay: 88.00  
 Deductions: 120.96  
 Net Pay: 167.04

**For assessment standards and criteria, see Assessment Checklist: Object-oriented Programming 2, INF3220–1.**

**ASSESSMENT CHECKLIST: DYNAMIC DATA STRUCTURES 1**

**INF3230-1**

STUDENT: \_\_\_\_\_

<b>STANDARD</b>	Students working <b>at standard</b> must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
-----------------	---

At Standard	<i>Dynamic Data Structures 1</i>			
<b>2</b>	<p><b><u>Problem-solving Phase:</u></b> The student:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> <li><input type="checkbox"/> identifies the appropriate use of linked lists required to solve the problem</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul> </td> </tr> </table>		<ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> <li><input type="checkbox"/> identifies the appropriate use of linked lists required to solve the problem</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> <li><input type="checkbox"/> identifies the appropriate use of linked lists required to solve the problem</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>			
<b>3</b>	<p><b><u>Implementation Phase:</u></b> The student creates a minimum of three programs containing the following—see sample assignment for Dynamic Data Structures 1:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification to existing classes are identified (object oriented only)</li> <li>• characteristics to be contained or inherited by new classes are identified (object oriented only)</li> <li>• pointers and linked lists are created as required</li> <li>• data is loaded into linked lists as required</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of stored data in arrays</li> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> </ul> </li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• data is transferred to objects (object oriented only)</li> <li>• sorting based on differing criteria (object oriented only)</li> <li>• search and merge routines (object oriented only)</li> <li>• dynamic memory is allocated and de-allocated as required</li> <li>• pointers are used to process data and direct program flow</li> <li>• data is processed in linked lists as required</li> <li>• nodes are added and/or deleted in linked lists as required</li> <li>• linked lists are traversed/searched/sorted</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• de-allocates dynamic memory</li> <li>• disposes of linked lists</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user's guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> </ul> </li> </ul> </td> </tr> </table>		<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification to existing classes are identified (object oriented only)</li> <li>• characteristics to be contained or inherited by new classes are identified (object oriented only)</li> <li>• pointers and linked lists are created as required</li> <li>• data is loaded into linked lists as required</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of stored data in arrays</li> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• data is transferred to objects (object oriented only)</li> <li>• sorting based on differing criteria (object oriented only)</li> <li>• search and merge routines (object oriented only)</li> <li>• dynamic memory is allocated and de-allocated as required</li> <li>• pointers are used to process data and direct program flow</li> <li>• data is processed in linked lists as required</li> <li>• nodes are added and/or deleted in linked lists as required</li> <li>• linked lists are traversed/searched/sorted</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• de-allocates dynamic memory</li> <li>• disposes of linked lists</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user's guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification to existing classes are identified (object oriented only)</li> <li>• characteristics to be contained or inherited by new classes are identified (object oriented only)</li> <li>• pointers and linked lists are created as required</li> <li>• data is loaded into linked lists as required</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of stored data in arrays</li> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• data is transferred to objects (object oriented only)</li> <li>• sorting based on differing criteria (object oriented only)</li> <li>• search and merge routines (object oriented only)</li> <li>• dynamic memory is allocated and de-allocated as required</li> <li>• pointers are used to process data and direct program flow</li> <li>• data is processed in linked lists as required</li> <li>• nodes are added and/or deleted in linked lists as required</li> <li>• linked lists are traversed/searched/sorted</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• de-allocates dynamic memory</li> <li>• disposes of linked lists</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user's guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> </ul> </li> </ul>			

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	---	--	---	--	---

Your school has several copy machines. The staff in the main office use one of the large ones. Normally, jobs on this machine are done in the order they are received. Usually the first job in is the first job done; however, rush jobs can be inserted into the list.

Design and code a program that uses a linked list to keep track of the order in which jobs must be done. Have the program randomly generate between five and ten duplicating jobs. Give each job a submit time and a completion time. Assume that these jobs are all submitted between 9:00 and 10:00 in the morning. Assume that each job takes 30 minutes. As each job is generated, it is added to a linked list that expands to accommodate the number of jobs. Once all jobs are added to the list, ask the user if he/she has any rush jobs that must be added to the list. If the answer is yes, ask the user for a “required by time.” Have the program insert this new job into the list so that it is completed prior to its “required by time” with the minimum amount of “bumping” of jobs already listed. Use “After 10:00” as a submitted time. Use a 24-hour clock.

Your program should generate the following output:

The following jobs were submitted for duplication:

Job 1: Submit Time 9:00	Projected Completion Time 9:30
Job 2: Submit Time 9:05	Projected Completion Time 10:00
Job 3: Submit Time 9:10	Projected Completion Time 10:30
Job 4: Submit Time 9:15	Projected Completion Time 11:00
Job 5: Submit Time 9:20	Projected Completion Time 11:30
Job 6: Submit Time 9:30	Projected Completion Time 12:00

Do you have any rush jobs (Y or N)? Y

What is the required by time for the job (use 24-hour notation)? 10:30

Your job has been assigned number 7.

The jobs were done in the following order:

Job 1: Submit Time 9:00	Projected Completion Time 9:30
Job 2: Submit Time 9:05	Projected Completion Time 10:00
Job 7: Submit Time After 10:00	Projected Completion Time 10:30
Job 3: Submit Time 9:10	Projected Completion Time 11:00
Job 4: Submit Time 9:15	Projected Completion Time 11:30
Job 5: Submit Time 9:20	Projected Completion Time 12:00
Job 6: Submit Time 9:30	Projected Completion Time 12:30

**For assessment standards and criteria, see Assessment Checklist: Dynamic Data Structures 1, INF3230–1.**

**ASSESSMENT CHECKLIST: DYNAMIC DATA STRUCTURES 2**

**INF3240-1**

STUDENT: \_\_\_\_\_

<b>STANDARD</b>	Students working <b>at standard</b> must demonstrate use of problem-solving techniques when producing programs, using criteria as noted in the checklists below. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
-----------------	---

At Standard	<i>Dynamic Data Structures 2</i>		
<b>2</b>	<p><b><u>Problem-solving Phase:</u></b> The student:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> <li><input type="checkbox"/> identifies the appropriate dynamic data structure required to solve the problem</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> <li><input type="checkbox"/> identifies the appropriate dynamic data structure required to solve the problem</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> defines the nature of the problem and outlines what the program must do</li> <li><input type="checkbox"/> creates an algorithm that identifies the initial state (initialization) input, processes, output and final state (termination) of the projected program</li> <li><input type="checkbox"/> identifies the appropriate constants, variables, derived data types, subprograms (functions and/or procedures), classes and class hierarchies required in the program</li> <li><input type="checkbox"/> identifies the appropriate dynamic data structure required to solve the problem</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> codes the algorithm, using a programming language</li> <li><input type="checkbox"/> documents comments to programmers</li> <li><input type="checkbox"/> debugs and tests sample data</li> <li><input type="checkbox"/> codes and formats the program properly</li> <li><input type="checkbox"/> evaluates the final product to insure proper implementation (see below)</li> </ul>		
<b>3</b>	<p><b><u>Implementation Phase:</u></b> The student creates a minimum of three programs containing the following—see sample assignment for Dynamic Data Structures 2:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification to existing classes are identified (object oriented only)</li> <li>• characters to be inherited by new classes are identified (object oriented only)</li> <li>• pointers, linked lists, stacks, queues and trees are created as required</li> <li>• data is loaded into linked lists, stacks, queues and trees as required</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of stored data in arrays</li> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> </ul> </li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• data is transferred to objects (object oriented only)</li> <li>• sorting based on differing criteria (object oriented only)</li> <li>• search and merge routines (object oriented only)</li> <li>• dynamic memory is allocated and de-allocated as required</li> <li>• pointers are used to process data and direct program flow</li> <li>• push and pop data</li> <li>• enqueue and dequeue data</li> <li>• data is processed in linked lists, stacks, queues and trees as required</li> <li>• nodes are added and/or deleted in linked lists and trees as required</li> <li>• linked lists and trees are traversed/searched/sorted</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• de-allocates dynamic memory</li> <li>• disposes of linked lists</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user's guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> </ul> </li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification to existing classes are identified (object oriented only)</li> <li>• characters to be inherited by new classes are identified (object oriented only)</li> <li>• pointers, linked lists, stacks, queues and trees are created as required</li> <li>• data is loaded into linked lists, stacks, queues and trees as required</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of stored data in arrays</li> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• data is transferred to objects (object oriented only)</li> <li>• sorting based on differing criteria (object oriented only)</li> <li>• search and merge routines (object oriented only)</li> <li>• dynamic memory is allocated and de-allocated as required</li> <li>• pointers are used to process data and direct program flow</li> <li>• push and pop data</li> <li>• enqueue and dequeue data</li> <li>• data is processed in linked lists, stacks, queues and trees as required</li> <li>• nodes are added and/or deleted in linked lists and trees as required</li> <li>• linked lists and trees are traversed/searched/sorted</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• de-allocates dynamic memory</li> <li>• disposes of linked lists</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user's guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Initialization and Input</b> <ul style="list-style-type: none"> <li>• stringed, integer and real variables</li> <li>• numeric and string constants</li> <li>• data entered through assignment statements and keyboard entry</li> <li>• appropriate local and global variables</li> <li>• data is stored in appropriate derived data types</li> <li>• error trapping occurs, using appropriate derived data types</li> <li>• data components of a class are identified (object oriented only)</li> <li>• modification to existing classes are identified (object oriented only)</li> <li>• characters to be inherited by new classes are identified (object oriented only)</li> <li>• pointers, linked lists, stacks, queues and trees are created as required</li> <li>• data is loaded into linked lists, stacks, queues and trees as required</li> </ul> </li> <li><input type="checkbox"/> <b>Processes</b> <ul style="list-style-type: none"> <li>• addition, subtraction, multiplication, division</li> <li>• predetermine, pre-check and post-check looping constructs</li> <li>• decision-making constructs</li> <li>• appropriate subprogram structures</li> <li>• proper one- and two-way parameter passing</li> <li>• summation of stored data in arrays</li> <li>• predefined string functions and procedures</li> <li>• methods to be used in classes are identified (object oriented only)</li> <li>• objects are constructed, employing user-defined classes (object oriented only)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Processes (continued)</b> <ul style="list-style-type: none"> <li>• data is transferred to objects (object oriented only)</li> <li>• sorting based on differing criteria (object oriented only)</li> <li>• search and merge routines (object oriented only)</li> <li>• dynamic memory is allocated and de-allocated as required</li> <li>• pointers are used to process data and direct program flow</li> <li>• push and pop data</li> <li>• enqueue and dequeue data</li> <li>• data is processed in linked lists, stacks, queues and trees as required</li> <li>• nodes are added and/or deleted in linked lists and trees as required</li> <li>• linked lists and trees are traversed/searched/sorted</li> </ul> </li> <li><input type="checkbox"/> <b>Output and Termination</b> <ul style="list-style-type: none"> <li>• rounds to a prescribed number of decimal places</li> <li>• lines up decimal points and inserts dollar signs where appropriate</li> <li>• column formatting occurs</li> <li>• de-allocates dynamic memory</li> <li>• disposes of linked lists</li> </ul> </li> <li><input type="checkbox"/> <b>Documentation and Presentation</b> <ul style="list-style-type: none"> <li>• presents statement of problem and algorithm to show how program was created</li> <li>• presents user's guide with clear and concise instructions</li> <li>• describes problems encountered during production and testing</li> <li>• aesthetic presentation: uses acceptable design principles</li> </ul> </li> </ul>		

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	---	--	---	--	---

Your school has several copy machines. The staff in the main office use one of the large ones. Normally, jobs on this machine are queued. Usually the first job in is the first job done; however, jobs can be prioritized. Rush jobs are done before ordinary jobs.

Design and code a program that uses a linked list to be used as a queue. Have the program randomly generate between five and ten duplicating jobs; give each job a submit time and a priority. As each job is generated, it is added to a linked list that expands to accommodate the number of jobs. Once all jobs are added to the queue, the program reorders the list so that rush jobs are done before ordinary jobs. Rush jobs are done in the order of submission. Use a 24-hour clock.

Your program should generate the following output:

The following jobs were submitted for duplication:

Job 1: Submit Time 9:00 Priority Normal  
Job 2: Submit Time 9:10 Priority Rush  
Job 3: Submit Time 10:20 Priority Normal  
Job 4: Submit Time 11:10 Priority Normal  
Job 5: Submit Time 12:00 Priority Rush  
Job 6: Submit Time 13:20 Priority Normal

The jobs were done in the following order:

Job 2: Submit Time 9:10 Priority Rush  
Job 5: Submit Time 12:00 Priority Rush  
Job 1: Submit Time 9:00 Priority Normal  
Job 3: Submit Time 10:20 Priority Normal  
Job 4: Submit Time 11:10 Priority Normal  
Job 6: Submit Time 13:20 Priority Normal

**For assessment standards and criteria, see Assessment Checklist: Dynamic Data Structures 2, INF3240–1.**

STUDENT: \_\_\_\_\_

<b>STANDARD:</b>	Students working <b>at standard</b> must demonstrate a post-secondary entry level understanding of the nature, approaches, areas of interest and algorithmic basis of the discipline of computer science. As the main focus of computer science is the utilization of algorithmic approaches to problem solving, most of the emphasis of this course is on the ability to understand design, develop, implement and test algorithmic solutions to problems amenable to intermediate modular and introductory object-oriented programming approaches. As this course is designed to be taught in conjunction with Programming 5 and Object-oriented Programming 1, this assessment checklist dovetails with the checklists for those courses. The column to the left of each checklist indicates the minimum rating for <b>at standard</b> performance. The rating scale at the bottom defines the different levels of competencies.
------------------	---

At Standard	<i>Computer Science 3</i>		
<b>2</b>	<p><b><u>Background Knowledge and Skills:</u></b> The student:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of the historical roots, processes, trends and general nature of the information revolution and the emerging information society</li> <li><input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding object-oriented design and programming approaches, such as object identification, analysis, creation and manipulation to the structured and modular approaches developed at earlier levels. The student now constructs algorithms and programs that reflect the OOD/OOP paradigm, which in turn employs structured and modular approaches</li> <li><input type="checkbox"/> demonstrates a general knowledge of core OOD/OOP concepts, such as encapsulation, inheritance and polymorphism</li> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, advantages and disadvantages of recursion and recursive processes</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of different types of files</li> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of Abstract Data Types and an ability to use both simpler data structures and classes to create Abstract Data Structures</li> <li><input type="checkbox"/> demonstrates an understanding of how a Turing machine can be used as a model of a general computing agent and used to explore the nature of problem analysis. As part of this process, students would construct and execute a number of standard algorithms such as a bit inverter or a parity checker on a Turing machine. This would likely best be done through the manipulation of an actual Turing machine. This machine could either be a computer simulation, a non-computer simulation or a paper-and-pencil depiction of a Turing machine</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of the historical roots, processes, trends and general nature of the information revolution and the emerging information society</li> <li><input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding object-oriented design and programming approaches, such as object identification, analysis, creation and manipulation to the structured and modular approaches developed at earlier levels. The student now constructs algorithms and programs that reflect the OOD/OOP paradigm, which in turn employs structured and modular approaches</li> <li><input type="checkbox"/> demonstrates a general knowledge of core OOD/OOP concepts, such as encapsulation, inheritance and polymorphism</li> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, advantages and disadvantages of recursion and recursive processes</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of different types of files</li> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of Abstract Data Types and an ability to use both simpler data structures and classes to create Abstract Data Structures</li> <li><input type="checkbox"/> demonstrates an understanding of how a Turing machine can be used as a model of a general computing agent and used to explore the nature of problem analysis. As part of this process, students would construct and execute a number of standard algorithms such as a bit inverter or a parity checker on a Turing machine. This would likely best be done through the manipulation of an actual Turing machine. This machine could either be a computer simulation, a non-computer simulation or a paper-and-pencil depiction of a Turing machine</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates an understanding of the historical roots, processes, trends and general nature of the information revolution and the emerging information society</li> <li><input type="checkbox"/> demonstrates a growing understanding of algorithmic problem solving by adding object-oriented design and programming approaches, such as object identification, analysis, creation and manipulation to the structured and modular approaches developed at earlier levels. The student now constructs algorithms and programs that reflect the OOD/OOP paradigm, which in turn employs structured and modular approaches</li> <li><input type="checkbox"/> demonstrates a general knowledge of core OOD/OOP concepts, such as encapsulation, inheritance and polymorphism</li> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, advantages and disadvantages of recursion and recursive processes</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of different types of files</li> <li><input type="checkbox"/> demonstrates a general understanding of the basic nature, creation and utility of Abstract Data Types and an ability to use both simpler data structures and classes to create Abstract Data Structures</li> <li><input type="checkbox"/> demonstrates an understanding of how a Turing machine can be used as a model of a general computing agent and used to explore the nature of problem analysis. As part of this process, students would construct and execute a number of standard algorithms such as a bit inverter or a parity checker on a Turing machine. This would likely best be done through the manipulation of an actual Turing machine. This machine could either be a computer simulation, a non-computer simulation or a paper-and-pencil depiction of a Turing machine</li> </ul>		
<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Design/Development Stage:</u></b> The student designs an algorithmic solution to a problem that employs the tactics developed in the Computer Science 1 and Computer Science 2 courses and in addition:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to use a simplified requirement analysis to cast a problem into a system of interacting objects</li> <li><input type="checkbox"/> demonstrates the ability to use appropriate class design approaches, such as iterative approaches to conceptualize these objects as composed of member functions or methods and data members or properties</li> <li><input type="checkbox"/> uses modular and structured approaches to outline the logic and structure of these object members employing program decomposition to the point where known idioms can be employed</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> addresses issues of abstraction, encapsulation and data hiding</li> <li><input type="checkbox"/> incorporates simple recursive approaches where appropriate</li> <li><input type="checkbox"/> incorporates data warehousing techniques through the use of files</li> <li><input type="checkbox"/> uses an appropriate construction technique, such as a simplified version of UML to create the class, object and activity diagrams needed to state the behaviour and properties of each object and the interaction among the objects</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to use a simplified requirement analysis to cast a problem into a system of interacting objects</li> <li><input type="checkbox"/> demonstrates the ability to use appropriate class design approaches, such as iterative approaches to conceptualize these objects as composed of member functions or methods and data members or properties</li> <li><input type="checkbox"/> uses modular and structured approaches to outline the logic and structure of these object members employing program decomposition to the point where known idioms can be employed</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> addresses issues of abstraction, encapsulation and data hiding</li> <li><input type="checkbox"/> incorporates simple recursive approaches where appropriate</li> <li><input type="checkbox"/> incorporates data warehousing techniques through the use of files</li> <li><input type="checkbox"/> uses an appropriate construction technique, such as a simplified version of UML to create the class, object and activity diagrams needed to state the behaviour and properties of each object and the interaction among the objects</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> demonstrates the ability to use a simplified requirement analysis to cast a problem into a system of interacting objects</li> <li><input type="checkbox"/> demonstrates the ability to use appropriate class design approaches, such as iterative approaches to conceptualize these objects as composed of member functions or methods and data members or properties</li> <li><input type="checkbox"/> uses modular and structured approaches to outline the logic and structure of these object members employing program decomposition to the point where known idioms can be employed</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> addresses issues of abstraction, encapsulation and data hiding</li> <li><input type="checkbox"/> incorporates simple recursive approaches where appropriate</li> <li><input type="checkbox"/> incorporates data warehousing techniques through the use of files</li> <li><input type="checkbox"/> uses an appropriate construction technique, such as a simplified version of UML to create the class, object and activity diagrams needed to state the behaviour and properties of each object and the interaction among the objects</li> </ul>		
<b>2</b>	<p><b><u>Utilization of Knowledge and Skills: Implementation Phase:</u></b> The student converts an algorithm into a program that employs all of the tactics developed in the Computer Science 1 and Computer Science 2 courses and in addition should:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> use a coding approach, such as iterative prototyping or the recursive/parallel approach that creates an initial core of functionality that is tested before being expanded to the next level. Ultimately this code/test cycle implements the entire algorithm</li> <li><input type="checkbox"/> encapsulate the properties and behaviours of the abstractions outlined in the algorithm into well structured classes and objects (ADTs)</li> <li><input type="checkbox"/> create and elaborates class hierarchies and libraries</li> </ul> </td> <td style="width: 50%; border: none; vertical-align: top;"> <ul style="list-style-type: none"> <li><input type="checkbox"/> create appropriate messaging mechanisms between client and server objects</li> <li><input type="checkbox"/> implement user interface based on objects</li> <li><input type="checkbox"/> use simple recursive approaches where appropriate</li> <li><input type="checkbox"/> incorporate appropriate file creation, reading, manipulation and writing techniques where required</li> </ul> </td> </tr> </table>	<ul style="list-style-type: none"> <li><input type="checkbox"/> use a coding approach, such as iterative prototyping or the recursive/parallel approach that creates an initial core of functionality that is tested before being expanded to the next level. Ultimately this code/test cycle implements the entire algorithm</li> <li><input type="checkbox"/> encapsulate the properties and behaviours of the abstractions outlined in the algorithm into well structured classes and objects (ADTs)</li> <li><input type="checkbox"/> create and elaborates class hierarchies and libraries</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> create appropriate messaging mechanisms between client and server objects</li> <li><input type="checkbox"/> implement user interface based on objects</li> <li><input type="checkbox"/> use simple recursive approaches where appropriate</li> <li><input type="checkbox"/> incorporate appropriate file creation, reading, manipulation and writing techniques where required</li> </ul>
<ul style="list-style-type: none"> <li><input type="checkbox"/> use a coding approach, such as iterative prototyping or the recursive/parallel approach that creates an initial core of functionality that is tested before being expanded to the next level. Ultimately this code/test cycle implements the entire algorithm</li> <li><input type="checkbox"/> encapsulate the properties and behaviours of the abstractions outlined in the algorithm into well structured classes and objects (ADTs)</li> <li><input type="checkbox"/> create and elaborates class hierarchies and libraries</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> create appropriate messaging mechanisms between client and server objects</li> <li><input type="checkbox"/> implement user interface based on objects</li> <li><input type="checkbox"/> use simple recursive approaches where appropriate</li> <li><input type="checkbox"/> incorporate appropriate file creation, reading, manipulation and writing techniques where required</li> </ul>		

<b>2</b>	<p><b>Utilization of Knowledge and Skills: Execution Phase:</b>                  The student tests and completes the documentation of a program created earlier by the student from an algorithm. These processes should employ all of the tactics developed in the Computer Science 1 and Computer Science 2 courses and in addition:</p> <table style="width: 100%; border: none;"> <tr> <td style="vertical-align: top;"> <p><input type="checkbox"/> <b>Testing</b></p> <ul style="list-style-type: none"> <li>• program is installed, configured and executed on a workstation with a different directory structure than the workstation used to code the program</li> </ul> <p><input type="checkbox"/> <b>Implementation</b></p> <ul style="list-style-type: none"> <li>• end user is given appropriate installation and configuration as well as execution instructions</li> </ul> </td> <td style="vertical-align: top; padding-left: 20px;"> <p><input type="checkbox"/> <b>Maintenance</b></p> <ul style="list-style-type: none"> <li>• user feedback is used to identify, design and code a revision to the program</li> </ul> <p><input type="checkbox"/> <b>Documentation</b></p> <ul style="list-style-type: none"> <li>• documents the contents of any class libraries to the point where a different programmer could use them in his or her program</li> </ul> </td> </tr> </table>	<p><input type="checkbox"/> <b>Testing</b></p> <ul style="list-style-type: none"> <li>• program is installed, configured and executed on a workstation with a different directory structure than the workstation used to code the program</li> </ul> <p><input type="checkbox"/> <b>Implementation</b></p> <ul style="list-style-type: none"> <li>• end user is given appropriate installation and configuration as well as execution instructions</li> </ul>	<p><input type="checkbox"/> <b>Maintenance</b></p> <ul style="list-style-type: none"> <li>• user feedback is used to identify, design and code a revision to the program</li> </ul> <p><input type="checkbox"/> <b>Documentation</b></p> <ul style="list-style-type: none"> <li>• documents the contents of any class libraries to the point where a different programmer could use them in his or her program</li> </ul>
<p><input type="checkbox"/> <b>Testing</b></p> <ul style="list-style-type: none"> <li>• program is installed, configured and executed on a workstation with a different directory structure than the workstation used to code the program</li> </ul> <p><input type="checkbox"/> <b>Implementation</b></p> <ul style="list-style-type: none"> <li>• end user is given appropriate installation and configuration as well as execution instructions</li> </ul>	<p><input type="checkbox"/> <b>Maintenance</b></p> <ul style="list-style-type: none"> <li>• user feedback is used to identify, design and code a revision to the program</li> </ul> <p><input type="checkbox"/> <b>Documentation</b></p> <ul style="list-style-type: none"> <li>• documents the contents of any class libraries to the point where a different programmer could use them in his or her program</li> </ul>		

<b>Rating Scale</b>	<b>4</b> – Demonstrates initiative that exceeds required knowledge/techniques/skills.	<b>3</b> – Consistently demonstrates all designated techniques/skills; rarely needs prompting.	<b>2</b> – Demonstrates all designated techniques/skills; occasionally needs prompting.	<b>1</b> – Demonstrates most designated techniques/skills; frequently needs prompting.	<b>0</b> – Does not demonstrate designated techniques/skills.
---------------------	---	--	---	--	---

**An Ecology Simulation (An enhanced version of the sample assignment outlined in INF\_CSSAMP1 and INF\_CSSAMP2)**

A biology teacher at your school has approached you to write a simple ecology simulation for an introductory science class. The ecology is to have the following characteristics:

- the setting for the ecology is to be a closed system, such as an island
- the ecology is to have one type of plant life, one type of herbivore and one type of carnivore
- the students (the users) will be allowed to set the size of the island (in hectares), the initial number of plants, the initial number of herbivores and the initial number of carnivores
- the simulation is to run in yearly cycles; during each year, the carnivores are to prey on the herbivores and the herbivores are to feed on the plant life. In addition, each species is to reproduce over the course of the year
- each carnivore requires 50 herbivores a year to survive, each herbivore requires 5000 plants a year to survive and each hectare of land can support up to 100 000 plants
- the carnivores' rate of reproduction is 1 to 2 (i.e., 1 carnivore can produce 2 offspring—assuming that there are a minimum of 2 carnivores in the system), the herbivores' rate of reproduction is 1 to 6 and the plants' rate of reproduction is 1 to 50
- assume that each species only consumes what it needs to survive (i.e., that the carnivores only eat 50 herbivores a year), that plants and animals that cannot get the food they need to survive die, that all calculations are done at the end of each year and that predation occurs prior to reproduction
- run the simulation for 10 years, or until the user wishes to exit or until the ecology “crashes”. (This ecology can be said to have crashed when one or more of the species dies out. Note: Simple ecologies are very unstable and are prone to crashing.)
- display the results of each cycle as a row in a table. As students are expected to be familiar with arrays and/or vectors, this data should be stored in an appropriate derived data type, such as an array, vector, record or struc and recalled when needed
- store the data in an appropriate file for future recall
- display the results as either a line graph or bar chart. The data for this display would have to be extracted from a derived data type, such as an array
- allow the user to interrogate the data to find information, such as the year of maximum carnivore population or the herbivore population for a specific year.

Design and develop an algorithm that:

- employs problem parsing to do a requirement analysis of the problem to identify the required classes and objects
- provides a description of the relationship that exists among the objects
- provides a generalized description of the methods and properties of each class
- uses a top–down, step-wise refinement approach to add specificity to each class
- uses an appropriate nomenclature, such as CRC Cards and Object Diagrams to outline the algorithm
- sketches a screen display(s) that illustrates the user interface and the format of the graphical output
- passes a walk-through or failed-on-paper test.

Translate the algorithm into an executable program that:

- maintains the logic and structure of the algorithm
- employs good object-oriented programming practices—loosely coupled, highly cohesive objects employing appropriate levels of encapsulation and data hiding
- incorporates the user interface and displays designed in the analysis and design stages
- uses a technique, such as iterative prototyping or parallel/recursive approaches to create and expand the core of functional code
- uses a combination of top–down coding and bottom–up coding:
  - to create the interface between individual objects
  - to create the objects
  - to test the objects
- makes appropriate use of derived data types to input, process and output information
- makes appropriate use of files to warehouse information
- has adequate internal and external documentation.

Test, implement and maintain the program by:

- executing the program with data known to produce specific output
- executing the program with aberrant data
- checking for congruency with the original requirements of the problem
- beta testing the program with knowledgeable users, such as another class member
- providing adequate installation, configuring and execution instructions to the end-user
- incorporating some user feedback in a revision of the program.

**For assessment standards and criteria, see Assessment Checklist: Computer Science 3, INF3210–1.**