The midterm summative task will have a report format using the following guidelines:

1. You will be asked to report on how the work you have done in the course so far relates to a number of topics.
2. For each topic, structure your report as follows.
   1. Summarize the work or activity you did that links to the topic.
   2. Explain specifically how the work or activity is related to the topic
   3. Provide or explain specific examples of your work
   4. Some topics may require you to develop some program code
3. You will have some choice regarding the topics you choose to answer.
4. Plan to answer at least 7 topics.
5. The task / report will be completed in two days (Tuesday & Wednesday)
6. You can plan ahead
7. The topic list on the following pages is in rough format but can help you to prepare. They are very detailed and we may have not covered all parts of the topics.
8. The final topics will be based on the rough topics. Some ideas will be combined and the wording will be simplified / improved

SEE THE TOPICS LIST ON THE NEXT FEW PAGES

**Topics – Rough Guidelines**

A1. use data structures in the design and creation of computer programs;

A1.3 describe the structure of one-dimensional

and two-dimensional arrays and related concepts

including elements, indexes, and bounds.

A2. demonstrate the ability to use standard algorithms in the design and creation of computer programs;

A2.2 demonstrate the ability to read from, and

write to, an external file *(e.g., sequential file,*

*database, XML file, relational database via SQL)*;

A2.3 demonstrate the ability to declare, initialize,

modify, and access one-dimensional

and two-dimensional arrays and elements

within a program;

A2.4 demonstrate the ability to add, change, or delete

elements of an array of objects in a program;

A3. demonstrate an understanding of object-oriented programming concepts and practices in the design and creation of computer programs;

A3.1 explain the importance of designing reusable

code in computer programs;

A3.2 explain fundamental object-oriented programming

concepts *(e.g., classes, objects, methods);*

A3.3 apply the concepts of scope and visibility for

variables, constants, and methods when creating

classes in computer programs;

A4. create clear and accurate internal and external documentation to ensure the maintainability of computer software.

A4.2 create clear and maintainable external user

documentation *(e.g., Help file, how-to manual,*

*FAQ, installation procedures, backup and recovery*

*procedures, training materials)* as part of a

complete software development project;

B1. design standard algorithms according to specifications;

B1.1 design algorithms to solve practical mathematical

problems *(e.g., amount of paint or carpet*

*needed, number of shingles needed, energy costs,*

*amount of water needed for an aquarium, projection*

*of Aboriginal youth population growth)*;

B2. design software solutions using object-oriented programming concepts;

B2.1 demonstrate the ability to create and use

instance methods *(e.g., constructors, mutators,*

*accessors)* in a computer program;

B2.2 design a simple base class to represent

objects or concepts in a problem statement,

using program templates or skeletons;

B2.3 write methods that require parameter

passing in a computer program.

B3. design user-friendly graphical user interfaces (GUIs) that meet user requirements;

B3.1 design graphical user interfaces that contain

common controls *(e.g., buttons, labels, text boxes)*;

B3.4 design responses to user events in a graphical

user interface.

B4. participate in a large student-managed project, using proper project management tools and techniques to manage the process effectively.

B4.2 create a project plan for a software development

project, outlining the tasks at each phase

of the software development life cycle;

B4.4 use industry-standard programming tools

*(e.g., UML [Unified Modeling Language], diagrams,*

*structure charts, flow charts, pseudocode)*

to develop a software project.

C1. demonstrate the ability to use project management tools to plan and track activities for a software development project;

C1.1 use software tools *(e.g., email, wikis, blogs,*

*task lists, bulletin boards, spreadsheets, shared*

*calendars)* to plan and track activities during a

software development project;

C1.2 communicate information about project

status *(e.g., completed, in progress, not started,*

*problems encountered, recommended modification*

*to deadlines)* effectively in writing

throughout the project.

C2. demonstrate the ability to use software development tools to design and write a computer program.

C2.1 use the features of a software development

environment to debug programs and create

functioning computer programs;

C2.2 work independently, using the Help function,

to resolve syntax issues while programming;

C2.3 work independently, using reference materials

*(e.g., code snippets, sample programs, APIs,*

*tutorials)*, to design and write functioning computer

programs.