COMP 2012H Assessment Rubrics

Course Learning Outcome	Exemplary	Competent	Needs Work	Unsatisfactory
1. Use common software tools to develop and debug a program written in an OOP language.	Use an IDE such as VS Code proficiently to write, compile, run, and debug a C++ program consisting of one or many source files.	Use an IDE such as VS Code effectively to write, compile, run, and debug a C++ program consisting of one or many source files.	Use an IDE such as VS Code to write, compile, and run a C++ program consisting of one source file. Have difficulty in dealing with programs consisting of more than one source file as well as debugging.	Have difficulty in using an IDE such as VS Code to write, compile, run, and debug a C++ program consisting of one source file without guidance.
2. Demonstrate that recursive and non-recursive functions are abstractions of sub-problems in a task.	Demonstrate thorough understanding of how recursion works. Be able to develop a recursive solution to a written problem on one's own, and sometimes contrast it with the corresponding non-recursive solution.	Demonstrate sufficient understanding of how recursion works. Be able to develop a recursive solution to a written problem if given the recursive algorithm, and sometimes contrast it with the corresponding non-recursive solution.	Demonstrate insufficient understanding of how recursion works. Be able to develop recursive solutions only to some simple problems, and only if the recursive algorithm is given. Cannot contrast the recursive solution with the corresponding non-recursive solution.	Is unable to understand how recursion works. Is unable to develop recursive solutions to problems even if the recursive algorithm is given.
3. Describe the concept and the use of pointers in indirect addressing and dynamic memory allocation.	Demonstrate strong understanding of the concept of pointers. Is able to use pointers effectively in indirect addressing and dynamic memory allocation in a great variety of scenarios.	Demonstrate sufficient understanding of the concept of pointers. Is able use pointers effectively in indirect addressing and dynamic memory allocation in standard scenarios.	Demonstrate marginal understanding of the concept of pointers. Is able to use pointers in indirect addressing and dynamic memory allocation only in simple scenarios.	Demonstrate little understanding of the concept of pointers. Have great difficulty in using pointers in indirect addressing and dynamic memory allocation even in simple scenarios.
4. Write object-oriented programs in C++ with object creation, destruction, member variables and functions, inheritance, polymorphism, and templates.	Demonstrates a comprehensive grasp of object-oriented concepts and fully demonstrates how to write object-oriented programs in C++.	Demonstrates a thorough grasp of object-oriented concepts and mostly demonstrates how to write object-oriented programs in C++.	Demonstrates a basic grasp of object-oriented concepts and barely able to demonstrate how to write object-oriented programs in C++.	Demonstrates a lack of grasp of object-oriented concepts and fail to demonstrate how to write object-oriented programs in C++.

Course Learning Outcome	Exemplary	Competent	Needs Work	Unsatisfactory
5. Analyze a program and provide simple solutions with OOP.	Demonstrates an exemplary ability to analyze programs written in OOP.	Demonstrates a proficient ability to analyze programs written in OOP.	Demonstrates a developing ability to analyze programs written in OOP.	Demonstrates deficiencies in their ability to analyze programs written in OOP.
6. Write basic algorithms associated with data structures such as stacks, queues, lists, trees, and hashes.	Demonstrates ability to fully implement algorithms with common data structures.	Demonstrates sufficient ability to implement algorithms with common data structures.	Demonstrates preliminary understanding of how to write algorithms with common data structures.	Demonstrates a lack of understanding of how to write algorithms with common data structures.
7. Define binary tree and search tree and describe how they are used to solve problems.	Demonstrates complete understanding of binary tree / binary search tree and is able to accurately describe how they are used to solve problems.	Demonstrates sufficient understanding of binary tree / binary search tree and successfully describe how they are used to solve problems.	Demonstrates preliminary understanding of binary tree / binary search tree and barely able to describe how they are used to solve problems.	Demonstrates deficient understanding of binary tree / binary search tree and unable to describe how they are used to solve problems.
8. Develop a large program using separate compilation, good OOP design, and code reuse.	Demonstrates complete comprehension of OOP concepts and techniques for the development of large programs.	Demonstrates basic comprehension of OOP concepts and techniques for the development of large programs.	Demonstrates minimal comprehension of OOP concepts and techniques for the development of large programs.	Demonstrates no comprehension of OOP concepts and techniques for the development of large programs.