Course Code	Course Title
COMP 2011	**Programming with C++**

Course Description

This course covers programming and data structures using C++. In addition to basic programming concepts such as variables and control statements, students will learn about arrays, pointers, dynamic data allocation, linked lists, stacks, queues, binary trees, recursion, and the basics of object oriented programming. Prerequisite(s): COMP 1021 OR COMP 1022P OR COMP 1022Q (prior to 2020-21) OR ISOM 3230. Exclusion(s): COMP 2012H

List of Topics

1. Introduction to computer programming
2. C++ basics: basic syntax, data types, operators
3. Control flow
4. Functions
5. Array and structure
6. Recursion
7. Scope
8. Struct
9. Pointers
10. Dynamic Data
11. Class
12. Stack and Queue
13. File input / output

Textbooks

Big C++: Late Objects, 3rd Edition
Cay S. Horstmann eBook ISBN: 9781119402978

Reference books
N/A

Grading Scheme

Lab exercises	10%
Programming assignments	24% (8%, 8% , 8%)
Quiz on C++ Basics	5%
Midterm	25%

Final exam	36%
Total	100%

Course Intended Learning Outcomes

1. Use common software tools to develop and debug a program written in C++.
2. Write and analyse short programs that solve simple problems in C++.
3. Demonstrate that recursive and non-recursive functions are abstractions of subproblems in a task.
4. Understand and demonstrate the use of pointers in indirect addressing and dynamic memory allocation.
5. Understand and demonstrate the use of various data structures.
6. Implement an abstract data type by defining a class in C++.

Assessment Rubrics

Course Learning Outcome	**Exemplary**	**Competent**	**Needs Work**	**Unsatisfactory**
Use common software tools to develop and debug a program written in C++.	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.

Write and analyze short programs that solve simple problems in C++.	Construct a solution to a written problem by writing a complete C++ program of no more than 500 lines of codes on one's own.	Construct a solution to a written problem by writing a complete C++ program of no more than 500 lines of codes on one's own if the requirements are clearly explained and the required programming constructs are told.	Require assistance to break down a written problem into sub-problems of sufficiently small sizes before being able to construct a solution for each sub-problem with no more than 100 lines of codes. The required programming constructs also need to be told.	Is unable to construct a solution to a written problem by writing a complete C++ program even under guidance. Skeleton code need to be given which breaks down the solution into a set of small functions with clear interface so that the student may be able to implement them.
Demonstrate that recursive and nonrecursive functions are abstractions of subproblems 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 nonrecursive 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.
Understand and demonstrate 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 pointerseffectively 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 inusing pointers in indirect addressing and dynamic memory allocation even in simple scenarios.

Understand and demonstrate the use of various data structures.	Is able to choose the appropriate data structures such as linked lists, stacks and queues to solve problems, and	Is able to use the required data structures such as linked lists, stacks and queues to solve problems, and implement the	Is able to use the required data structures such as linked lists, stacks and queues to solve simple problems, and	Demonstrate little understanding of data structures such as linked lists, stacks and queues, and is unable to use them to solve
	implement the solution in C++ on one's own.	solution in C++ on one's own.	implement the solution in C++ with guidance.	problems even with guidance.
Implement an abstract data type (ADT) by defining a class in C++.	Given the description of a simple ADT, is able to implement it with a complete C++ class definition that includes appropriate class members and class member functions. Is able to write programs that create and manipulate ADT objects.	Given the description of a simple ADT, able to implement it with a complete C++ class definition when its class members and class member functions are also hinted. Is able to write programs that create and manipulate ADT objects.	Given the description of the C++ class definition of a simple ADT, including its class members and class member functions, is able to implement the member functions. Sometimes is able to write programs that create and manipulate ADT objects.	Have great difficulty in understanding the link between a simple ADT and its C++ definition. Is unable to complete C++ definition for simple ADTs and to program with C++ classes.