Course Code	Course Title
COMP 2012	**Object-Oriented Programming and Data Structures**

Course Description

To learn the fundamental concepts and techniques behind object-oriented programming. They include: abstract data types; creation, initialization, and destruction of objects; class hierarchies; polymorphism, inheritance and dynamic binding; generic programming using templates. To learn the object-oriented view of data structures: linked lists, stacks, queues, binary trees, and algorithms such as searching and hashing. Prerequisite(s): COMP 2011 Exclusion(s): COMP 2012H

List of Topics

1. Revision of dynamic data structures
2. C++ class basics
3. Separate compilation and makefile
4. Constructors, destructor, initialization
5. Inheritance, polymorphism, and dynamic binding
6. Generic programming
7. STL: containers, iterators, algorithms
8. Namespace
9. Static member functions/data
10. Hashing
11. Binary search trees
12. AVL trees
13. rvalue reference and move semantics

Textbooks & Reference books

Paul Deitel, Deitel & Associates (2017). C++ How to Program

M.A. Weiss (2014). Data Structures and Algorithm Analysis in C++.

Data Structures and Algorithm Analysis Ed. 3.2 (C++ Version).

B. Eckel (2000). Thinking in C++.

L. Nyhoff (2005). ADTs, Data Structures and Problem Solving with C++.

Stanley Lippman (2013). C++ Primer.

Grading Scheme

Programming assignment – PA1	9%
Programming assignment – PA2	9%
Programming assignment – PA3	9%
Laboratory exercises	10%
Midterm	25%
Final examination	38%
Total	100%

Course Intended Learning Outcomes

1. Write object-oriented programs in C++ with object creation, destruction, member variables and functions, inheritance, polymorphism, and templates.
2. Analyze simple problems and provide solutions with OOP.
3. Understand the basic operations of data structures such as stacks, queues, lists, binary search trees, and hashes, and their implementations.
4. Demonstrate the ability to use the learned data structures to solve problems in C++.
5. Develop large programs using separate compilation, good OOP design, and code reuse through the use of inheritance, generic programming and the standard template library.

Assessment Rubrics

Course Learning Outcome	**Exemplary**	**Competent**	**Needs Work**	**Unsatisfactory**
1. Write object-oriented programs in C++ with object creation, destruction, member variables and functions, inheritance, polymorphism, and templates.	Demonstrate a comprehensive grasp of object-oriented concepts and fully demonstrates how to write object-oriented programs in C++.	Demonstrate a thorough grasp of object-oriented concepts and mostly demonstrates how to write object-oriented programs in C++.	Demonstrate a basic grasp of object-oriented concepts and barely able to demonstrate how to write object-oriented programs in C++.	Demonstrate 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
2. Analyze simple problems and provide solutions with OOP.	Demonstrate an exemplary ability to analyze problems and solve them using OOP.	Demonstrate a proficient ability to analyze problems and solve them using OOP.	Demonstrate a developing ability to analyze problems and solve them using OOP.	Demonstrate deficiencies in their ability to analyze problems and solve them using OOP.
3. Understand the basic operations of data structures such as stacks, queues, lists, binary search trees, and hashes, and their implementations.	Demonstrate excellent understanding of the basic operations of the learned data structures and ability to implement them.	Demonstrate sufficient understanding of the basic operations of the learned data structures and ability to implement them.	Demonstrate limited understanding of the basic operations of the learned data structures and limited ability to implement them.	Demonstrate a lack of understanding of the basic operations of the learned data structures and inability to implement them.
4. Demonstrate the ability to use the learned data structures to solve problems in C++	Demonstrate an exemplary ability to use the learned data structures to solve problems in C++.	Demonstrate a proficient ability to use the learned data structures to solve problems in C++.	Demonstrate a developing ability to use the learned data structures to solve problems in C++.	Demonstrate deficiencies in the ability to use the learned data structures to solve problems in C++.
5. Develop large programs using separate compilation, good OOP design, and code reuse through the use of inheritance, generic programming and the standard template library.	Demonstrate complete comprehension of OOP concepts and techniques for the development of large programs.	Demonstrate basic comprehension of OOP concepts and techniques for the development of large programs.	Demonstrate minimal comprehension of OOP concepts and techniques for the development of large programs.	Demonstrate no comprehension of OOP concepts and techniques for the development of large programs.