1. **Course Number and Course Title:**

   CMP 305 – Data Structures and Algorithms

2. **Credit Hours:**

   3 – 2 – 3

3. **Prerequisites and/or Co-Requisites:**

   Prerequisite: CMP 220 Programming II
   Prerequisite/concurrent: CMP 213 Discrete Structures or MTH 213 Discrete Mathematics
   Co-requisites: CMP 305L

4. **Name and Contact Information of Instructor:**

   Name: Dr. Michel Pasquier

5. **Course Description (Catalog Description):**

   Covers the design, analysis, and implementation of abstract data types and related algorithms to solve computing problems efficiently. Includes fundamental data structures such as arrays, linked lists, stacks, and queues, as well as advanced data structures such as trees, hash tables, heaps, and graphs. Studies algorithms for manipulating these data structures, recursive programming, searching, and sorting. Laboratory work includes substantial programming assignments.

6. **Textbook and other Supplemental Material:**

   Textbook:
   - M.A. Weiss, *Data Structures and Algorithms Analysis in C++,* 4th ed., Addison-Wesley/Pearson, 2014.

   Other supplemental material:
   - A. Drozdek, *Data Structures and Algorithms in C++,* 4th ed., Cengage, 2013.
   - T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms,* 3rd ed., MIT Press, 2009.
   - D.S. Malik, *C++ Programming: Program Design Including Data Structures,* 7th ed., Cengage, 2016.

7. **Course Learning Outcomes:**

   Upon completion of the course, students will be able to:
   1. Derive the time complexity of basic programs and algorithms to evaluate their performance when handling large amounts of data.
   2. Implement Abstract Data Types (ADTs) such as vectors, lists, stacks, queues, trees, heaps, priority queues, hash tables, and graphs.
   3. Develop programs that make use of one or more ADTs, using the C++ object-oriented programming language and the C++ Standard Library.
   4. Understand the use of recursion to process data structures and implement algorithms.
   5. Apply searching and sorting algorithms to solve computing problems effectively.

8. **Teaching and Learning Methodologies:**

   Methods include lectures, problem-based learning, class discussions, and group work, as well as laboratory sessions. Students learning is assessed via in-class quizzes, exams, homework, and programming assignments/projects.

9. Course Topics and Schedule:

Topic/Activity	Weeks
Introduction to data structures and algorithms	Week #1
Algorithm analysis, computational time complexity	Week #2
Abstract data types, containers, C++/STL library	Week #3
Linked lists, arrays, performance comparison	Week #4
Vectors and lists, swap/move, iterators	Week #5
Stacks and queues, adaptor classes	Week #6
Recursion, design and programming, applications	Week #7
Accumulator recursion, tail optimization – Midterm I	Week #8
Trees and graphs, traversals, binary trees	Week #9
Binary search trees, balanced trees, AVL trees	Week #10
Heaps, array-based binary heaps, priority queues	Week #11
Hash tables and associative arrays, hashing – Midterm II	Week #12
Sorting, strategies and algorithms	Week #13
Graphs, basic search algorithms	Week #14
Revision	Week #15
Final Exam	Week #16