1. **Course number and name**  
   **CSC 220: Data Structures**

2. **Credits and contact hours**  
   3 credits  
   Contact hours: 150 minutes of lecture sessions /week

3. **Instructor’s or course coordinator’s name**  
   Course coordinator: James Wong, Professor of Computer Science

4. **Text book, title, author, and year**  
   Data Structures and Abstractions with Java, Frank Carrano, current edition  
   a. **other supplemental materials**  
      Lecture slides

5. **Specific course information**  
   a. **brief description of the content of the course (catalog description)**  
      Linear and non-linear data structures, including lists, stacks, queues, trees, tables and graphs. Recursion, iteration over collections, sorting, searching, Big O notation and hash table.  
   b. **prerequisites or co-requisites**  
      grades of C or better in MATH 226 and CSC 210. MATH 227 may be taken concurrently.  
   c. **indicate whether a required, elective, or selected elective course in the program**  
      Required for Computer Engineering.

6. **Specific goals for the course**  
   a. **specific outcomes of instruction, ex. The student will be able to explain the significance of current research about a particular topic.**  
      At the end of this course students will  
      • Be able to write Java programs in an integrated development environment  
      • Utilize a debugger in software development  
      • Apply Data Structures and ADT concepts effectively when developing small to medium sized projects  
      • Write robust code utilizing exception handling language features  
      • Learn what and how to document each project  
   b. **explicitly indicate which of the student outcomes listed in Criterion 3 or any other outcomes are addressed by the course.**  
      Course addresses ABET Student Outcome(s): a, b, c, e, j, k.

7. **Brief list of topics to be covered**
• Object-oriented programming: Object-oriented design; encapsulation and information-hiding; separation of behavior and implementation; classes, subclasses, and inheritance; polymorphism; class hierarchies; collection classes and iteration protocols; generic types;
• Recursion: The concept of recursion; recursive specification of mathematical functions (such as factorial and Fibonacci); simple recursive procedures (Towers of Hanoi, permutations, fractal patterns); implementation of recursion
• Introduction to computational complexity: Asymptotic analysis of upper and average complexity bounds; big-O notation; standard complexity classes; empirical measurements of performance
• Fundamental computing algorithms: O(N log N) sorting algorithms (Quicksort, heapsort, mergesort); hashing, including collision-avoidance strategies;
• Abstraction and implementation of classic data structures: lists, stacks, queues, priority queues, hash tables, graphs, trees & balanced trees and dictionaries