1. **Course number and name**
   - **EE 2372 Software Design I**
2. **Credits and contact hours**
   - **3 credits, 3 contact hours**
3. **Instructor's or course coordinator's name**
   - **Michael McGarry**
4. **Text book, title, author, and year**
   - **Programming in C (4th Edition) by Stephen Kochan**
     a. **other supplemental materials**
        i. **Data Structures and Algorithms in C++ (2nd Edition) By Michael Goodrich, Roberto Tamassia, and David Mount**
5. **Specific course information**
   a. **brief description of the content of the course (catalog description)**
      i. **Foundations of data structures and algorithms. These foundations include: space and time complexity analysis, the use of data structures such as linked lists and binary trees, basic sorting and searching algorithms, and foundations of software testing/verification/validation.**
   b. **prerequisites or co-requisites**
      i. **Prerequisites: CS 1320 with a grade of "C" or better**
   c. **indicate whether a required, elective, or selected elective (as per Table 5-1) course in the program**
      i. **Required course**
6. **Specific goals for the course**
   a. **Become a proficient user of the Linux software development environment and GNU software development tool-chain**
      i. **Linux software development environment**
      ii. **GNU software development tools – *gcc, gdb, make, gprof, gcov***
   b. **Understand C language programming constructs**
      i. **variables, algebraic and logical expressions (including operator set)**
      ii. **simple I/O**
      iii. **decision statements, iterative control statements**
   c. **Understand and follow structured software design strategies [*ABET 2a/b*]**
      i. **programming paradigms: procedural/modular, object-oriented**
      ii. **design for reuse using the procedural/modular paradigm**
      iii. **utilizing standard libraries, focus on C standard library**
   d. **Understand and utilize fundamental data structures [*ABET 1c*]**
      i. **arrays and structures**
      ii. **strings and string processing**
      iii. **pointers, linked lists, and binary trees**
      iv. **storage allocation: static, stack and heap**
   e. **Software testing, verification, and validation**
      i. **Understand the differences between testing, verification, and validation.**
      ii. **Demonstrate an understanding of unit testing strategies and tradeoffs.**

iii. **Ability to construct test vectors and use tools to automate their construction.**

f. **Understand the foundations of algorithm analysis [*ABET 1a*]**
   i. **history and the role of algorithms**
   ii. **algorithms available in the C standard library**
   iii. **determine time complexity of algorithms**
   iv. **determine space complexity of algorithms**

g. **Understand and utilize fundamental algorithms [*ABET 1c*]**
   i. **sorting algorithms: bubble sort and insertion sort**
   ii. **searching algorithms: linear search, binary search, and hash functions**

h. **explicitly indicate which of the student outcomes listed in Criterion 3 or any other outcomes are addressed by the course.**
   i. **Student Outcome 1a and 1c, "an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics"**
   ii. **Student Outcome 2a and 2b, "an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors"**

7. **Brief list of topics to be covered**
   - **GNU/Linux software development environment**
   - **C language programming constructs: variables, algebraic expressions, simple I/O**
   - **C language programming constructs: decision statements and iterative control statements**
   - **Fundamental data structures: arrays, pointers, and structures**
   - **Fundamental data structures: strings and string processing**
   - **C standard library: console and file I/O**
   - **Software engineering process**
   - **Software testing/verification/validation**
   - **Object-oriented programming with C++**
   - **Fundamental data structures: linked-lists**
   - **Fundamental data structures: binary trees**
   - **Fundamental algorithms: sorting and searching**
   - **Time and space complexity analysis of algorithms**