1. **TITLE OF COURSE AND COURSE NUMBER:** Design and Analysis of Algorithms, CS372, Credits: 3, (Required Advanced Computer Science Course)

2. **DESCRIPTION OF THE COURSE:** An introduction to the concepts, methodologies, and constructive models for formulating algorithms. Use of analytic techniques to determine the relative efficiency of algorithms with respect to several measures such as time and space complexity. Later topics introduce alternate models of computation such as probabilistic algorithms, parallel processing, and complexity classes (such as NP).

3. **COURSE PREREQUISITES:** CS 342 with a grade of C- or better and Math 324

4. **COURSE OBJECTIVES:**
   - Introduce complexity measures and develop student's abilities in using Big-O concepts, notation, and combinatorial methodologies in the analysis of algorithms.
   - Teach fundamental standard design methods for algorithms.
   - Introduce complexity classes, classical P and NP issues.

5. **STUDENT LEARNING OUTCOMES:**
   Upon completion of this course, students will:
   a) Understand complexity theory sufficiently and have enough practice to conduct a Big-O(), -omega, and -theta analysis of a particular program/programming problem. They must demonstrate an appreciation of program design decision-making based upon their combinatorial, asymptotic, and local/pragmatic analyses. Proofs here will involve developing inequalities between functions and using calculus concepts such as limits and derivatives to derive/formulate growth behavior theorems. Students are expected to understand, formulate, and prove those results. This requires extensive problem-solving using all their calculus, probability, discrete structures, and programming background as a foundation. Several handouts of drill problems and proofs will be used in problems solving sessions and homeworks. Students are expected to solve 10-20 of them for each homework. Students are expected to also become adept at solving several categories of recurrence relations. Measure: exams and homework projects.
   b) Be capable of explaining rudimentary lower-bound theory and the intuitive and formal requirements, methodology, and approaches to a mathematical-proof in this area.
Measure: exams
c) Effectively and analytically apply several fundamental design methods for algorithms, demonstrating the ability to select methodologies, fully design the algorithms from them, and produce executing programs based on those design methodologies. At least one full program in each design methodology is required as projects. Measure: exams and homework projects.
d) Read the literature of the discipline and conduct inquiry-based studies of problems in the area (in the form of a required term paper and other research-based projects). This will be conducted using the library and Internet resources such as on-line books and research papers. Measure: projects.
e) Appreciate the historical development and current results of the area in terms of classic and recent breakthrough papers in each of the sub-disciplines. Measure: exams and projects.
f) Through classroom participation in problem-solving sessions and discussions, homework, papers, and other assignments, this course also reinforces the following student learning outcomes in accordance with the university:
   • Effectively and exactly express themselves in written and oral form. Measure: exams and projects.
   • Demonstrate the ability to think critically both in algorithmic synthesis and analysis. Measure: exams, surveys, and projects.
   • Locate and use information on the Internet, in monographs and scholarly journals. Measure: projects.
   • Demonstrate the ability to integrate knowledge and ideas in a coherent and meaningful manner (as cited above). Measure: exams, surveys, and projects.

6. TOPICAL OUTLINE OF THE COURSE CONTENT:
   1) Review algorithm specification
   2) Combinatorial analytic methods
   3) Recursion
   4) Design Methodology
      a. Divide and Conquer
      b. Greedy Methods
      c. Dynamic Programming
      d. Basic Search & Traversal Techniques
      e. Backtracking
      f. Algebraic Simplification & Transforms
   5) Rudiments of Lower Bound Theory
   6) NP Problems
   7) Probabilistic Algorithms
   8) Parallel Algorithms
   9) Techniques for Algorithm Verification

7. GUIDELINES/SUGGESTIONS FOR TEACHING METHODS AND STUDENT LEARNING ACTIVITIES:
a) Classroom lectures, discussions, and problem solving sessions  
b) Homework reviews  
c) Lab work with using compilers, profilers, & simulators  

8. GUIDELINES/SUGGESTIONS FOR METHODS OF STUDENT ASSESSMENT (OUTCOMES):  
a) Periodic examinations and final  
b) Programming Projects  
c) Term Paper  

9. SUGGESTED READINGS, TEXTS, OBJECTS OF STUDY:  
   Foundations of Algorithms (Using Java Pseudocode), Richard Neapolitan & Kumarss Naimipour, Jones and Bartlett Publishers, 2004  
   Several handouts & papers.  

10. BIBLIOGRAPHY OF SUPPORTIVE TEXTS AND OTHER MATERIALS:  
   Aho, Hopcroft, & Ullman, Design and Analysis of Computer Algorithms, Addison-Wesley, 1974  
   Baase, Sara & Allen van Gelder, Computer Algorithms (3rd Edition), Addison-Wesley, 2000  
   Bentley, Jon, More Programming Pearls, Addison-Wesley, 1988  
   Lewis, Harry R., and Larry Denenberg, Data Structures and Their Algorithms,
Addison-Wesley, 1991.

11. **PREPARER'S NAME AND DATE**: Dr. John Najarian

12. **ORIGINAL DEPARTMENTAL APPROVAL DATE**: Spring, 1996


14. **DEPARTMENTAL REVISION APPROVAL DATE**: Spring 2005