OC11B: C Programming: Algorithms and Techniques

Course Description
Programming in C: Algorithms and Techniques is the second course in our C programming sequence. Students in this course learn a little more advanced concepts of C including functions, arrays and sorting algorithms. The course begins with the introduction to functions-what are functions, the need of functions and it's usefulness in top-down designing of programs, how to pass parameters to the functions, how to return values from function. Then it introduces arrays-what are arrays, how are arrays stored in the memory, how to access array cells. It then introduces how to pass arrays to the functions. During the course work, we insist students to follow the rules of structured programming so that they learn how to write clear and efficient programs along with the new concepts. It then introduces the typedef data type and recursion. There is a final project in the course which makes the student exercise everything learned in the course.

Required Text
The Art and Science of C,
by Eric Roberts

Course Topics
Programming in C- Algorithms and Techniques requires that students master the following specific topics:

- **Functions in C**
  - What are functions?
  - The importance of functions.
  - Functions in C – Fundamental concepts.
  - Function prototypes.
  - Writing the body of the function.
  - Invoking the function Actual and formal parameters.
  - The program as a whole.

- **Arrays**
  - What are arrays?
  - Representation of arrays and the size of operator
  - Passing arrays to functions Multidimensional arrays

- **Data types**
  - User defined types – typedef

- **Library functions**
  - User defined Functions
  - Execution with stack frames
  - The binary search algorithm

- **Computational Complexity**
  - Running time of linear search, binary search, selection sort
Recursion
- Introduction—The Towers of Hanoi
- stack frames

Learning Objectives
Upon completion of in C: Algorithms and Techniques, students will demonstrate the ability to:
- design and implement solutions to problems by writing, running, and debugging computer programs.
- use and implement commonly used algorithms and data structures.
- develop and select appropriate algorithms to solve problems.
- understand the basic concepts used to analyze the performance of algorithms.

Sample Assignment
Write a program that removes all duplicate values from a sorted array of integers, leaving only a single copy of each value. For example if the input array is inpArray with the following values,

- **InpArray**
  - 50 60 70 72 80 85 85 90 100
  
  then the output array will be,

- **OutArray**
  - 50 60 70 72 80 85 90 100

Your program should include a function named removedup that returns the new size of the array (without the duplicates).

In addition, your program should display both the input array and the output array. File name should be, removeD.c

In the Fibonacci series the first two terms are 0 and 1 and every subsequent term is the sum of the two preceding terms. So the first 7 elements of the series are the following:

- \( F_0 = 0 \) \( F_1 = 1 \)
- \( F_2 = F_0 + F_1 \) \( F_3 = 2 (F_1 + F_2) \) \( F_4 = 3 (F_2 + F_3) \) \( F_5 = 5 (F_3 + F_4) \) \( F_6 = 8 (F_4 + F_5) \)
- \( F_7 = 13 (F_5 + F_6) \)

Write a recursive function (fib(n) that returns the nth Fibonacci number. DO NOT use any loops here. The program should read n from the user and then call the fib function to compute the value.

Grading Policy
A numerical course grade will be calculated according to the following formula:

- Software Exercises: 10%
- Programming Assignments: 25%
- Project: 15%
- Final Exam: 50%