BHARATI VIDYAPEETH'S
INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT (BVICAM)
(Affiliated to Guru Gobind Singh Indraprastha University, Approved by AICTE, New Delhi)
A-4, Paschim Vihar, Rohtak Road, New Delhi-110063, Visit us at: http://www.bvicam.in/

Course Code: MCA-103 Course Name: Programming in C

Assignment 2

(Based on Unit-III)

Q 1. Suppose we wish to analyze a line of text by examining each of the characters and determining into which of several different categories it falls. In particular, we count the number of vowels, consonants, digits, whitespace characters and "other" characters (punctuation, operators, brackets, etc.) This can be accomplished by reading in a line of text, storing it in a one-dimensional character array, and then analyzing the individual array elements. An appropriate counter will be incremented for each character. The value of each counter (number of vowels, number of consonants, etc.) can then be written out after all of the characters have been analyzed. Design a complete C program that will carry out such an analysis. First define the following variables.

line = an 80-element character array containing the line of text

vowels = an integer counter indicating the number of vowels

consonants = an integer counter indicating the number of consonants

digits = an integer counter indicating the number of digits

whitespc = an integer counter indicating the number of whitespace characters (blank spaces or tabs)

other = an integer counter indicating the number of characters that do not fall into any of the preceding categories

Note that newline characters are not included in the "whitespace" category, because there can be no newline characters within a single line of text. Structure the program so that the line of text is read into the main portion of the program, and then passed to a function where it will be analyzed. The function will return the value of each counter after all of the characters have been analyzed. The results of the analysis (i.e., the value of each counter) will then be displayed from the main portion of the program.

Q 2. Consider the well-known problem of rearranging (i.e., sorting) a list of n integer quantities into a sequence of increasing values. Design a sorting program in such a manner that unnecessary storage will not be used. Therefore the program will contain only one dynamic array-a one-dimensional, integer array called x, which will be rearranged one

element at a time. The rearrangement will begin by scanning the entire array for the smallest number. This number will then be interchanged with the first number in the array, thus placing the smallest number at the top of the list. Next the remaining n - 1 numbers will be scanned for the smallest, which will be exchanged with the second number. The remaining n - 2 numbers will then be scanned for the smallest, which will be interchanged with the third number, and so on, until the entire array has been rearranged. The complete rearrangement will require a total of n - 1 passes through the array, though the length of each scan will become progressively smaller with each pass. In order to find the smallest number within each pass, sequentially compare each number in the array, x [i] ,with the starting number, x[item], where item is an integer variable that is used to identify a particular array element. If x[i] is smaller than x[item], then we interchange the two numbers; otherwise we leave the two numbers in their original positions. Once this procedure has been applied to the entire array, the first number in the array will be the smallest. Then repeat the entire procedure n -2 times, for a total of n - 1 passes (item =0, 1, . . .,n -2). The only remaining question is how the two numbers are actually interchanged. To carry out the interchange, first temporarily save the value of x[item] for future reference. Then assign the current value of x[i] to x[item]. Finally, assign the original value of x[item], which has temporarily been saved, to x[i].

Q 3. Determine for the version of C available on your particular computer, how many memory cells are required to store a single character? An integer quantity? A long integer? A floating-point quantity? A double-precision quantity? Explain?

Q 4. What is the purpose of the indirection operator? To what type of operand must the indirection operator be applied?

Q 5. Can the address operator act upon an arithmetic expression, such as 2 * (U + v)? Explain the reasons for your answer?