**Labsheet 7**

**Strings Library Functions**

1. Write a program which reads in two strings and compares them using the function **strcmp().** Print out a message describing whether or not the two strings are the same. Recall that **strcmp()** returns **0** if the strings are the same.
2. Read in two strings from the user. Use the string library functions to combine them into a single string. Print out the two original strings and the result of the concatenation.
3. Write a program to read in a string and display the count of vowels in the string.
4. Write your own version of the string library functions **strlen( ),** **strcpy( )**.

**Functions & Arrays**

1. Type in, compile, and run the following program.

**#include <stdio.h>**

**int abs( int arg )**

**{**

**if ( arg < 0 )**

**return –arg;**

**else**

**return arg;**

**}**

**main()**

**{**

**int inval;**

**printf(“Enter an integer: “);**

**scanf(“%d”, &inval);**

**printf(“Absolute value is: %d\n”, abs(inval) );**

**}**

1. Modify the previous program so that abs() works for **float** values instead of **int**’s. Change **main()** so it uses the new function.
2. The ***prefix sums*** of an array are the values found by summing all elements of the array from the beginning to the current element. The first prefix sum is just the first element of the array, the second prefix sum is the sum of the first two elements, the third is the sum of the first three elements and so on. An array and its prefix sums are shown below:

**Array: 1 2 3 4 5**

**Prefix Sums: 1 3 6 10 15**

Write a function which computes the prefix sums of an array. The function needs 3 arguments: the array containing the data, an array in which to store the prefix sums it computes, and the length of the arrays. The prototype for the function should look like this:

**void PrefixSums ( int data[], int result[], int length);**

1. Write a program to search for a particular key in an integer array using Linear Search.
2. Write a program to search for a particular key in an integer array using Binary Search.
3. Write a function which sorts an array of integers into ascending order. The function should take two arguments: the array and an integer specifying the length of the array. Write a main program which fills an array with random values using **rand()** and **srand()** and then calls the sort function to order the values in the array. Print the contents of the array both before and after the call to the sort function.
4. When passing multidimensional arrays as parameters to a function, every dimension but the first one must be specified. For example, a function which takes a  **2 X 2** array of integers as an argument would be declared like this:

**void f( int a[][2] )**

**{**

**/\* ….. \*/**

**}**

This is required because multidimensional arrays are stored in one contiguous block of memory and the compiler needs to know the length of the rows in order to correctly calculate the location of the element to be accessed.

Write functions which implement matrix addition, subtraction, multiplication. The function will take three **2 X 2** arrays of integers as arguments (call them **A, B and C).** The function should add the corresponding elements of **A** and **B** and store the results in **C.**

1. Write a program to Read and Display a 3-D array.

**Recursion**

1. Write a recursive function to to find the factorial of any number.
2. Write a recursive function to print (a) all numbers from 1 to N

(b) all numbers from N to 1.

1. Write a recursive function to find the sum of all numbers from 1 to N.
2. Write a recursive function to find the sum of all numbers an integer array.
3. Write a recursive function to find the *nth* Fibonacci number.
4. Write a recursive version of the pow() function .