**Homework Assignment -1**

1. Solve the following sub-questions on pointers.
   1. Write a function to swap two integers using pointers.
   2. Write a function that takes three variable (a, b, c) in as separate parameters and rotates the values stored so that value a goes to b, b to c, and c to a.
   3. Write a function that uses pointers to initialize an array of integers to zero. Return the array using a pointer (**int \*zero\_array(int \*a, int length)**)
   4. Write a program to compute the sum of elements in an array using pointers.
   5. Write a program to read through an array of any type using pointers. Write a C program to scan through this array to find a particular value.
2. Write a program for each of the following sub-questions.
   1. Compute the length of a string using pointers (**int strlen(char \*str)**)
   2. Copy a string from one location to another (**strcpy(char \*src, char \*dest)**)
   3. Concatenate two strings ‘s’ and ‘t’ (**char \*strcat(char \*s, char \*t)**)
   4. Returns 1 if the string *t* occurs at the end of string *s*, and otherwise returns 0

(**int strend(char \**s,* char *\*t*)**)

* 1. Print if the given string is a palindrome or not (**void checkPalindrome(char \*s)**)
  2. Reverse a given string (**char \*reverse(char \*original)**)

1. Write a program to find the number of times that a given word(i.e. a short string) occurs in a sentence (i.e. a long string!). Read data from standard input. The first line is a single word, which is followed by general text on the second line. Read both up to a newline character, and insert a terminating null before processing.

Typical output should be:

The word is "the".

The sentence is "the cat sat on the mat".

The word occurs 2 times.

1. Write a program evaluate\_Polish which evaluates a reverse Polish expression (read about this notation if you don’t already know) from the command line, where each operator or operand is a separate argument. For example:

evaluate\_Polish 3 5 6 + \*

evaluates to

3 \* (5 + 6)

1. Given a string of letters, you have to count the frequency of occurrence of each character in the string.

To make it more rigid on memory, follow a few guidelines:  
a) Initially no memory is allocated for any of the letters.  
b) When you read a character, in case the character has already occurred in the string before, you just have to simply increase the count of that character in the array.  
c) When you read a character, in case the character is occurring in the string for the first time, you have to allocate a structure "node" dynamically with the "element" field initialized to the current character and the "count" field initialized to 1.

Your task is to print the final array of character counts after simulating for the whole string in the above mentioned way.

A sample input is provided for more clarification(s).

**Input:**  
The input will consist of only one line made up of a string of characters. The length of the string will not exceed 100 characters and it will consist of upper case or lower case letters only.  
**Output:**  
The array of character(s) with their count(s) in the order they are allocated. Print only the characters that occur in the string.  Print in each line, a character and its frequency.

Make sure you don’t use any extra memory!  
  
**Sample Input:**  
ccbbbbaaccaaz

**Sample Output:**  
c 4  
b 4  
a 4  
z 1