Lab 6 Quizzes and Answers

- Write a c program which prints an input string from end to beginning.

Sample Scenario:

Enter a string : cankayauniversity
Original format : cankayauniversity
Converted format : ytisverinuayaknac

```c
#include <stdio.h>

int main(void)
{
    /* input variable is used to hold the user input string */
    char input[50];
    /* i variable is used to print a specific character in user input */
    int i=0;

    /* First we take the input string from user */
    printf("Enter a string :");
    scanf("%s", &input);

    /* We print each character to the screen from beginning up to the
    null character
    Null character always exists at the end of the string. */
    printf("Original Format :");
    while(input[i] != '\0')
    {
        printf("\%c", input[i]);
        i++;
    }
    printf("\n");

    /*
    Now the value of i is equals to the length of the input string
    So decrementing index by one and printing each character
    will print the input string from end to begininning.
    */
    printf("Converted Format :");
    while( i >= 0)
    {
        printf("\%c", input[i]);
        i--;
    }
    printf("\n");

    return(0);
}
```
- Write a C program which takes an input and prints “PALINDROME” if the input can be read the same way in either direction.

Sample Scenario:

nalan
PALINDROME
emre
NOT A PALINDROME

#include <stdio.h>

int main(void)
{
    /* input variable is used to hold the user input string */
    char input[150]=""
    /* i is used for the for loop iterations */
    int i=0;
    /* length represents the total length of the input string*/
    int length=0;
    /* total equality shows the number of equal characters */
    int totalEquality = 0;
    /* neccessary equality represents the number of equal characters
to be a palindrme
    for the input string.*/
    int neccessaryEquality = 0;
    /* First we take the input string from user */
    printf("Enter a string : ");
    scanf("%s", &input);
    /* First we should determine the length of the input word. */
    while(input[length] != '\0')
    {
        length++;
    }
    if(length % 2 == 0)
    {
        /* if the word has number of even characters,
           length / 2 characters must be equal.
           So, there is no middle character.
       */

        /*
        if
        */

        /*
        */

    }
neccessaryEqualiy = length / 2;

/*
we compare all the characters in either directions.
*/
for(i=0 ; i< neccessaryEquality; i++)
{
    /*
    if the reverse characters are same,
then we increment total equality variable
*/
    if(input[i] == input[length - 1 - i])
    {
        totalEquality ++;
    }
}
else
{
    /*
    if the word has number of odd characters,
we must eliminate the middle character
*/
    neccessaryEquality = (length - 1) / 2;

    /*
    we compare all the characters in either directions.
*/
    for(i=0 ; i< neccessaryEquality; i++)
    {
        /*
        if the reverse characters are same,
then we increment total equality variable
*/
        if(input[i] == input[length - 1 - i])
        {
            totalEquality ++;
        }
    }
}
if(totalEquality == neccessaryEquality)
    printf("PALINDROME\n");
else
    printf("NOT A PALINDROME\n");
return (0);
Define all core operators (+, -, *, /, %) in a character array and define two different integer arrays with size 3. Select a random operator from character array and apply that operator to all numbers which is at the same index and assign the results to a third integer array.

Sample Scenario:

Assume that * is selected randomly. Your integer arrays has the elements of 3, 6, 9 and 1, 2, 3. You should;
- multiple 3 by 1 (at the index zero)
- multiple 6 by 2 (at the index one)
- multiple 9 by 3 (at the index two)

You should use <stdlib.h> to use random generator built in function.

The code piece written below is given as hint.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    /* index variable is used to determine the operation which we will apply */
    int index = rand() % 5;
    /* we define all core operators */
    char op[5] = {'+', '-', '*', '/', '%'};
    /* we will make some operations on two user-defined arrays.
    We will use these array's elements as sources. */
    int a[3] = {3, 6, 9};
    int b[3] = {1, 2, 3};
    int c[3] = {0, 0, 0};

    /* third array will hold the results after the operations completed.
    Third array's elements is used as destination */
    int i = 0;

    /* we apply the selected operation to the same indexed elements of user-defined arrays and assign the result to the third array */
    switch(op[index])
    {
        case '+':
            c[0] = a[0] + b[0];
            c[1] = a[1] + b[1];
            break;
        case '-':
            c[0] = a[0] - b[0];
```
c[1] = a[1] - b[1];
break;

case '++':
    c[0] = a[0] * b[0];
    c[1] = a[1] * b[1];
break;

case '/':
    c[0] = a[0] / b[0];
    c[1] = a[1] / b[1];
break;

case '%':
    c[0] = a[0] % b[0];
    c[1] = a[1] % b[1];
break;

default:
    printf("The software never reaches here");
break;
}

for (i=0; i<3; i++)
    printf("c[%d] = %d\n", i , c[i]);

return(0);
- Define an integer array which has 6 elements with unordered list. Your program should sort these integers and assign this sorted list to another array. At the end of the software print your new array with ordered form. Your array should have the elements of 8,3,9,4,1,6

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int a[6] = {8,3,9,4,1,6};
    int hold;
    int i=0;
    int arraySize = 6;

    for ( int pass = 0; pass < arraySize - 1; pass++ )
    {
        for ( i = 0; i < arraySize - 1; i++ )
        {
            /* if the previous element is greater than the next element we should change their indexes. */
            if ( a[ i ] > a[ i + 1 ] )
            {
                /* while changing the indexes we first hold the current element */
                hold = a[ i ];
                /* we assign next element's value to the current element's value. so current element is removed. */
                a[ i ] = a[ i + 1 ];
                /* after assigning the variable that we hold to the next index, we complate replacement process. */
                a[ i + 1 ] = hold;
            }
        }
    }

    for(i=0; i<arraySize; i++)
        printf("%d\n", a[i]);

    return(0);
}
```