**Data Structures and Algorithms**

**Lab 9**

**Submitted To:**

Mr. Dilshad Sabir

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

***Convert the following iterative function to a recursive one.***

**Complete the function *‘bool test\_palindrome\_itr(char \* test\_word)’* in the provided skeleton code.**

**Program:**In this program, function “***reverse\_num\_array”*** takes a numeric array entered by user and size of this array as its argument. This function implements a stack and pushes the entries of the numeric array into this stack through for loop. Again a for loop iterates and pops the elements of stack and stores them in a numeric array. Hence the elements of numeric array entered by the user are printed in reversed order on the console.



**Output:**



**Task 2:**

***Convert the following recursive function to iterative one.***

**Program:**In this program, a function ***“isBalanced”*** implements a stack and evaluates the expression entered by the user; if an opening bracket is encountered, it is pushed into the stack and flag is set to **‘1’**. If a respective closing bracket is encountered the previously pushed opening bracket is popped. When the complete expression has been evaluated and

stack is empty; message **“Expression is balanced”** is displayed on the console.

****

**Output:**



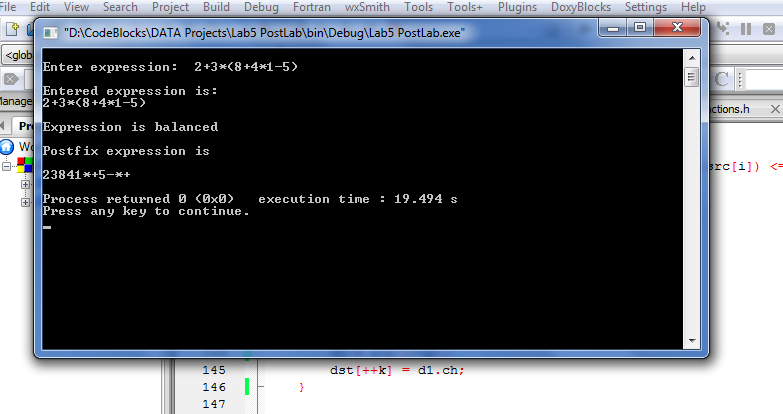
**Post Lab:**

***Write a program to reverse a string using recursion*.**

**Program:**In this program, function ***“char\* reverse(char\* str)”*** evaluates the expression entered by user. If the scanned character is operand, it is added to the output; brackets are pushed into stack, if found balanced they are popped respectively. Operators are also pushed into stack and popped according to their priority levels. At last, postfix expression is printed on the console.



**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE END**