Name: T Abishek Entry Number: 2019CS10407
Name: Gaurav Jain Entry Number: 2019CS10349

COL216: Assignment 2

Introduction:

This assignment aims to calculate the value of a postfix expression given as an input in the form of a string. A simple way to calculate the value of postfix expressions is using a stack to store the integers of range 0 to 9 appearing in the expression. If we encounter an operator (in this assignment, +, *, and -), we pop out the top two elements from the stack, perform the operation, and push the calculated value back into the stack. If the stack has fewer than two elements, then we throw an error.
A simple example to demonstrate the above expression:
Consider the postfix expression: *930*+*
Execution:
9 > push onto the stack. Stack contents: 9
3 > push onto the stack. Stack contents: 9 3
0 > push onto the stack. Stack contents: 9 3 0
* > pop the top 2 stack elements, perform the multiplication, then push the result onto the stack. Stack contents: 9 0
+ > pop the top 2 stack elements, perform the addition, then push the result onto the stack. Stack contents: 9
Output: 9

Approach to the code:

The user is asked to input the postfix expression string. The string is read character by character. If we encounter any invalid characters in the string, we raise an error and exit the program. Otherwise, if the character is of ASCII range 48 to 57, it is an integer of range 0 to 9, and thus, we store it in the stack.
Since in this assignment, we have only three operators: +, -, and *. We check for each operator separately by the ASCII values of these operators. Based on the operator we have encountered, we perform the proper operation and push the result back in the stack. To operate, we need two integers, and we take these integers from the top of the stack. But if the stack has less than two elements, the given postfix expression is incorrect. We raise an error in this case.
We keep on doing this process until we reach the end of the given string and output the only element left in the stack. If there is more than one element in the stack, we raise an error as the given expression is incorrect.
Suitable comments are also present inside the code for a better understanding of the code.

The input is taken as a postfix expression. The intermediate results are stored as 32 bit signed registers in the stack, so if the result of any intermediate result exceeds the range (-2147483648 to 2147483647), it will lead to an overflow error and will give a wrong result.

Our testing strategy involved thinking and figuring out what can happen when a user interacts with our program and then implementing the code to handle such cases. Cases like entering invalid characters, entering more numbers in the expression than the

operators require, entering more operators than the given numbers can take have all been handled while writing the program. The testing strategy also involved trying out the different scenarios using various test cases that test all the different scenarios that can transpire while running the program. The different kinds of test cases that were used are shown below. The result was also calculated manually and matched for checking the accuracy of the program.

Testcases:

1. Testcase 1-
   This testcase shows the result when only a single number is entered.
   Input-
   Please enter Postfix Expression, then press ENTER:
   4
   Output-
   Result: 4

---

2. Testcase 2-
   This testcase shows the error message when only an operator is entered.
   Input-
   Please enter Postfix Expression, then press ENTER:
   +
   Output-
   Incorrect Postfix Expression!

---

3. Testcase 3-
   This testcase shows the result when a valid expression is given with two numbers and one operator.
   Input-
   Please enter Postfix Expression, then press ENTER:
   45-
   Output-
   Result: -1

---

4. Testcase 4-
   This testcase shows the error message when an invalid expression is given, which has more than one element in the stack at the end of the process.
   Input-
   Please enter Postfix Expression, then press ENTER:
   15*67-
   Output-
   Incorrect Postfix Expression!

Name: T Abishek Entry Number: 2019CS10407
Name: Gaurav Jain Entry Number: 2019CS10349

5. Testcase 5-
   This testcase shows the error message when an invalid expression is given, which has an invalid character in the string.
   Input-
   Please enter Postfix Expression, then press ENTER:
   23/
   Output-
   Unexpected char in the Postfix string!

---

6. Testcase 6-
   This testcase shows the error message when an empty string is entered.
   Input-
   Please enter Postfix Expression, then press ENTER:

   Output-
   Incorrect Postfix Expression!

---

7. Testcase 7 -
   This testcase shows the error message when an invalid character is entered
   Input -
   Please enter Postfix Expression, then press ENTER:
   23a
   Output-
   Unexpected char in the Postfix string!

---

8. Testcase 8 -
   Input -
   Please enter Postfix Expression, then press ENTER:
   34*45*+
   Output-
   Result: 32

---

9. Testcase 9 -
   Input -
   Please enter Postfix Expression, then press ENTER:
   9784***
   Output-
   Result: 2016

---

10. Testcase 10 -
    Input -
    Please enter Postfix Expression, then press ENTER:
    45*965+--
    Output-
    Result: 22

Name: T Abishek Entry Number: 2019CS10407
Name: Gaurav Jain Entry Number: 2019CS10349

11. Testcase 11 -
    Input -
        Please enter Postfix Expression, then press ENTER:
        Result: 123*+987+-*
    Output-
        Result: -42

---

12. Testcase 12 -
    Input -
        Please enter Postfix Expression, then press ENTER:
        7412**-
    Output-
        Result: -1

---

13. Testcase 13 -
    Input -
        Please enter Postfix Expression, then press ENTER:
        1254**412*++-
    Output-
        Result: -45

---

14. Testcase 14 -
    Input -
        Please enter Postfix Expression, then press ENTER:
        4512569874512++*-*-+*+-*+
    Output-
        Result: -5376
