COL216: Assignment 2

Approach

1. Postfix expression is taken as input in one line in the form of string and stored in memory.
2. Maximum 1024 characters can be taken as input. If more given, computation will be done on first 1024 characters and warning " *Your input is too big, answer might be frivolous."* will be thrown.
3. Address of input string is loaded in $t0 and character at that address in $t1.
4. Equality check is done on $t1 and accordingly pop and push is done in stack.
5. If $t1 is digit between 0-9 then it is pushed in stack, and head pointer is changed accordingly.
6. If $t1 is operator +,-,* then appropriate is computation is done on last two numbers of stack, they are popped and the computation is pushed back.
7. Once $t1 becomes empty or "\n" then computation stops.
8. If any other value of $t1 is encountered then error "*INVALID-INPUT: Input is not in postfix"* will be thrown and computation stops.
9. Now that computation is done, our answer will be at head pointer of stack (if input was correct postfix expression) otherwise number stored at head is the partial computation of postfix subset of input. My code will print this partial computation at this step with a warning "*Your input is not complete postfix, ans above is partial answer."* Otherwise correct answer is printed.

Wrong Input

1. The case when input containing any character other than {0-9,+,-,*} is stated in approach section above.
2. The case when operators is less than (number of digits -1) is stated above, if it is more then (number of digits -1) then error "*INVALID-INPUT: Input is not in postfix"* will be thrown.
3. The case when last character is not an operator the code will throw error "*INVALID-INPUT: Input is not in postfix"*
4. The case when first two numbers are not digits, if only one number is input then it itself will be given as answer and if more than two characters are given and then first two are not digits then again error "*INVALID-INPUT: Input is not in postfix"* is thrown.

```
Enter the postfix expression: +2
INVALID-INPUT: Input is not in postfix

Enter the postfix expression: 2+2
INVALID-INPUT: Input is not in postfix

Enter the postfix expression: 2+22+
INVALID-INPUT: Input is not in postfix

Enter the postfix expression: 22+2*3-66+
Partial computation: 12

WARNING: Your input is not complete postfix, ans above is partial answer.
```

```
Enter the postfix expression: as2++
INVALID-INPUT: Input is not in postfix

Enter the postfix expression: 2
Computation of given postfix expression is: 2

Enter the postfix expression: 22+3++++
INVALID-INPUT: Input is not in postfix
```

Test Cases

1. Once my code was done, I first tested it with the testcase given in question. Code gave correct answer in that, then I used induction to prove the correctness of my code.
2. Proof by induction on number of digits in the expression.
    a. BASE CASE:
    If n = 1, only one number is given, then it itself is given as output.
    If n = 2, then 3 cases are possible, for 3 operators, code given correct answer for all 3 cases.

```
Enter the postfix expression: 54+
Computation of given postfix expression is: 9

Enter the postfix expression: 54-
Computation of given postfix expression is: 1

Enter the postfix expression: 54*
Computation of given postfix expression is: 20
```

If n = 3, then 9 cases are possible, for 3 operators, code given correct answer for all 3 cases.

```
Enter the postfix expression: 23+4-
Computation of given postfix expression is: 1

Enter the postfix expression: 234*+
Computation of given postfix expression is: 14

Enter the postfix expression: 23+4+
Computation of given postfix expression is: 9

Enter the postfix expression: 23-4-
Computation of given postfix expression is: -5

Enter the postfix expression: 234*-
Computation of given postfix expression is: -10
```

```
Enter the postfix expression: 23-4+
Computation of given postfix expression is: 3

Enter the postfix expression: 23*4-
Computation of given postfix expression is: 2

Enter the postfix expression: 23*4*
Computation of given postfix expression is: 24

Enter the postfix expression: 23*4+
Computation of given postfix expression is: 10
```

Hence code computes result of operation of two dogits correctly and correctly computes the final result if third digit is also given for all possible combinations.

b. INDUCTION HYPOTHESIS:
Let code works correctly for n = k >=4.

c. INDUCTION STEP:
Let n = k+1,
Hence expression = f(x) (+ or – or *) a ……a is some digit and n(f(x)) = k

Computed currectly from IH, let it b.

⇨ b (+ or – or *) a => this computed correctly from base case.

3. Test cases for rigorous testing:
   NOTE: these are generated by code given, and correct answer of original prefix is calculated from calculator. (since answer of both postfix and prefix is going to be same as both denotes same expression)

```
Enter the postfix expression: 325*+
Computation of given postfix expression is: 13

Enter the postfix expression: 209*+5-9+9-
Computation of given postfix expression is: -3

Enter the postfix expression: 209*+5-8+9-
Computation of given postfix expression is: -4

Enter the postfix expression: 494*-7+2-8+16*5*-
Computation of given postfix expression is: -49
```

Registors Used

t0 -> storing address of input string

t1 -> bit vise value

t2,t3 -> calculation in steps

t4 -> stack entries count

t5 -> counter

$s0 -> max value -> 1024