Experiment2.3

Student Name: Rohit Kumar Mahato **UID: 21BCS7480**
Branch: CSE **Section/Group: 717A**
Semester: 3 **Date of Performance:13/10/2022**
Subject Name: Data Structure **Subject Code:21CSH-211**

Aim:- Write a program to demonstrate the use of stack (implemented using linear array) in converting arithmetic expressions from infix notation to postfix notation.

Algorithm:-

- Start
- Scan the expression from left to right.
- Whenever the operands arrive, print them.

- infix = reverse(infix)
  loop i = 0 to infix.length
  if infix[i] is operand → prefix+= infix[i]
  else if infix[i] is '(' → stack.push(infix[i])
  else if infix[i] is ')' → pop and print the values of stack till the symbol ')' is not found
  else if infix[i] is an operator(+, -, *, /, ^) →

- **if** the stack is empty then push infix[i] on the top of the stack.
  Else →
  If precedence(infix[i] > precedence(stack.top))
  → Push infix[i] on the top of the stack
  else if(infix[i] == precedence(stack.top) && infix[i] == '^')
  → Pop and print the top values of the stack till the condition is **true**
  → Push infix[i] into the stack
- **else if**(infix[i] == precedence(stack.top))
  → Push infix[i] on to the stack
- Else **if**(infix[i] < precedence(stack.top))

- → Pop the stack values and print them till the stack is not empty and infix[i] < precedence(stack.top)
- → Push infix[i] on to the stack
- End loop
- Pop and print the remaining elements of the stack
- Prefix = reverse(prefix)
- Print the prefix in the terminal
- End

Code:-

```
#include<iostream>
#include<stack>
using namespace std;

int prec(char ch) {
    if (ch == '^')
        return 3;
    else if (ch == '/' || ch == '*')
        return 2;
    else if (ch == '+' || ch == '-')
        return 1;
    else
        return -1;
}

string infixToPostfix(string s) {

    stack<char> st;
    string ans = "";

    for (int i = 0; i < s.length(); i++) {
        char ch = s[i];


        if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0'
&& ch <= '9'))
            ans += ch;


        else if (ch == '(')
            st.push('(');


        else if (ch == ')') {
```

```
            while (st.top() != '(')
            {
                ans += st.top();
                st.pop();
            }
            st.pop();
        }


        else {
            while (!st.empty() && prec(s[i]) <= prec(st.top())) {
                ans += st.top();
                st.pop();
            }
            st.push(ch);
        }
    }
    while (!st.empty()) {
        ans += st.top();
        st.pop();
    }

    return ans;
}

int main() {
    string s;
    cin >> s;
    cout << infixToPostfix(s);
    return 0;
}
```

Output:-

```
PROBLEMS 1   OUTPUT   TERMINAL   DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\mahat\OneDrive\Desktop\vscode> cd "c:\Users\mahat\OneDrive\Desktop\vscode\Stacks\" ; if ($?)
}
m*n+(p-q)+r
mn*pq-+r+
PS C:\Users\mahat\OneDrive\Desktop\vscode\Stacks>
```

Learning Outcomes:

1. Understanding the concept of push, pop, peep, and Change.
2. Learn the concept of the conversation of infix to prefix.
3. Understanding the concept of the Method and Function.