Experiment - 2.3

Student Name: Rohan Jaiswal **UID:21BCS2856**
Branch: CSE **Section/Group: 608 B**
Semester: 3rd **Date of Performance:28th Oct 2022**
Subject Name: Data Structure **Subject Code:** 21CSH-211

Aim of the practical:

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

Algorithm:

Step 1: Initialize the Stack

Step 2: Scan the operator from left to right in the infix expression.

Step 3: If the leftmost character is an operand, set it as the current output to the Postfix string.

Step 4: And if the scanned character is the operator and the Stack is empty or contains the '(', ')' symbol, push the operator into the Stack.

Step 5: If the scanned operator has higher precedence than the existing **precedence** operator in the Stack or if the Stack is empty, put it on the Stack.

Step 6: If the scanned operator has lower precedence than the existing operator in the Stack, pop all the Stack operators. After that, push the scanned operator into the Stack.

Step 7: If the scanned character is a left bracket '(', push it into the Stack.

Step 8: If we encountered right bracket ')', pop the Stack and print all output string character until '(' is encountered and discard both the bracket.

Step 9: Repeat all steps from 2 to 8 until the infix expression is scanned.

Step 10: Print the Stack output.

Step 11: Pop and output all characters, including the operator, from the Stack until it is not empty.

Step 12: Stop.

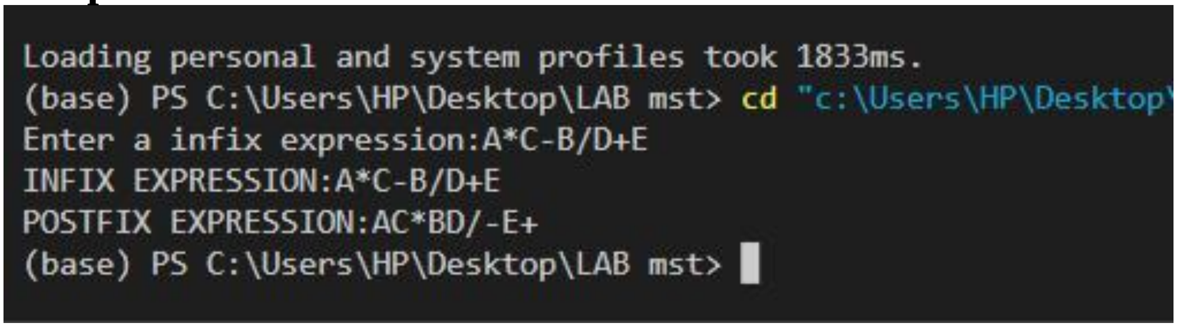
Program code:

```
#Include<bits/stdc++.>
using namespace std;
int prec(char c){
if(c=='^'){ return 3;
```

```
    }
     else if(c=='*' || c=='/'){
          return 2;
     }
     else if(c=='+' || c=='-'){
         return 1;
     }
     else{
         return -1;
     }
 }
 void infixToPostfix(string s){ stack<char> st; st.push('N'); int
   l=s.length(); string ns; for(int i=0;i<l;i++){ if((s[i]>='a' &&
   s[i]<='z') || (s[i]>='A' && s[i]<='Z')){
                    ns+=s[i];
        }
        else if(s[i]=='('){
                    st.push('(');
        }
        else if(s[i]==')'){

 while(st.top()!='N' && st.top()!='('){
                char c=st.top(); st.pop();
               ns+=c;
}
              if(st.top()=='('){ char
                c=st.top(); st.pop();
             }
        }
        else{ while(st.top()!='N' && prec(s[i])<=prec(st.top())){ char
          c=st.top(); st.pop();
 ns+=c;
           }
           st.push(s[i]);
        }
    }
  while(st.top()!='N'){ char
     c=st.top(); st.pop(); ns+=c;
    }
  cout<<ns<<endl;
}
int main(){ string n;
   cout<<"Enter the infix expression: "; cin>>n;
   cout<<"Postfix expression is: "<<endl;
 infixToPostfix(n);
   return 0;
}
```

Output:



```
Loading personal and system profiles took 1833ms.
(base) PS C:\Users\HP\Desktop\LAB mst> cd "c:\Users\HP\Desktop\
Enter a infix expression:A*C-B/D+E
INFIX EXPRESSION:A*C-B/D+E
POSTFIX EXPRESSION:AC*BD/-E+
(base) PS C:\Users\HP\Desktop\LAB mst>
```

Learning Outcomes:

1. I have learnt about the Stack.
2. I have learnt about stack operations.
3. I have learnt about infix to postfix in stack.
4. I have learnt about time complexity of stack.