Name of Student :- Pavan Ratan Gole

UID:- Na

Batch:- C4

Exp. No. 1

Aim:- To convert Infix Expression to PostFix expression. Output the content of the intermediate stack along with the postfix expression
Define the structure of the stack and stack pointer variable
PUSH and POP and PEEK Operations with overflow and under flow conditions taken care
Functions to check the operator's precedence and whether the given character is an operator or operand.

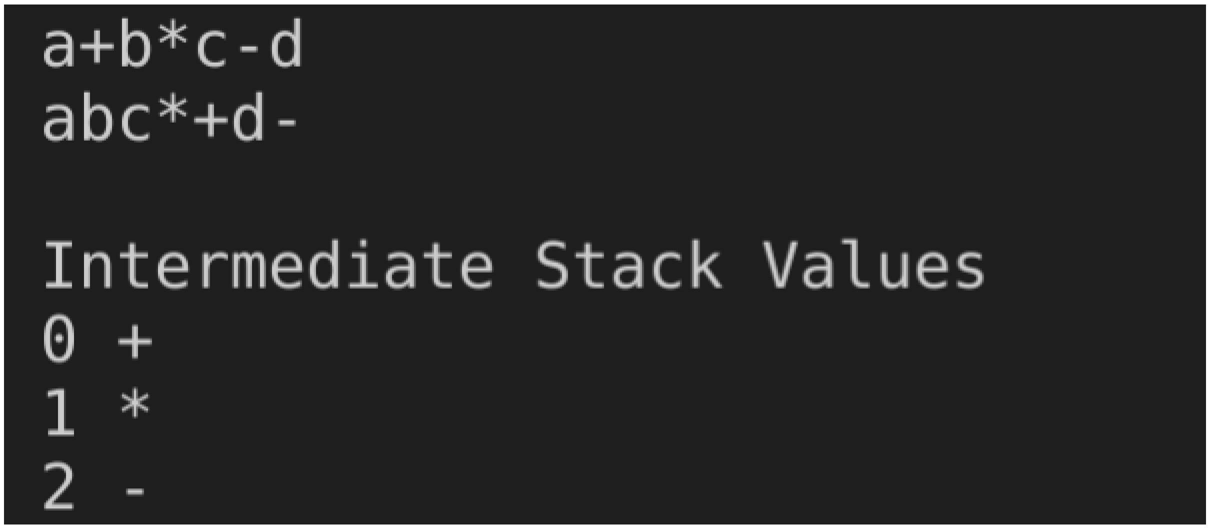
Program:-

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

class StackApp
{
private:
   /* data */
   int top = -1;
   char data[10];

public:
   bool isEmpty()
   {
       if (top != -1)
       {
           return false;
       }
       return true;
   }

   bool isFull()
   {
       if (top != 5)
```

```
        {
            return false;
        }
        return true;
    }

    int push(int item)
    {
        if (isFull())
        {
            cout << "Stack Overflowed" << endl;
            return 0;
        }
        ++top;
        data[top] = item;
        return 0;
    }

    int pop()
    {
        if (isEmpty())
        {
            return 0;
        }
        char temp = data[top];
        --top;
        return temp;
    }

    int peek()
    {
        if (isEmpty())
        {
            return -1;
        }
        return data[top];
    }
};

int precedence(char ope)
```

```
{
    if (ope == '(' || ope == ')')
    {
        return 3;
    }
    if (ope == '/' || ope == '*')
    {
        return 2;
    }
    return 1;
}

bool isOperator(char ope)
{
    if (ope == '+' || ope == '-' || ope == '/' || ope == '*' || ope == '('
|| ope == ')')
    {
        return true;
    }
    return false;
}

void infixTopostfix(char array[], int n)
{
    StackApp *stack = new StackApp();

    for (int i = 0; i < n; i++)
    {
        if (!isOperator(array[i]))
        {
            cout << (char) array[i];
        }
        else if (stack->isEmpty())
        {
            stack->push(array[i]);
        }
        else
        {
            if (precedence((char) stack->peek()) < (char)
precedence(array[i]))
            {
                stack->push(array[i]);
            }
            else
            {
                while (!stack->isEmpty())
                {
                    cout <<  (char) stack->pop();
                }
                stack->push(array[i]);
            }
        }
    }
    while (!stack->isEmpty())
    {
        cout << (char) stack->pop();
    }
    cout << endl;
}

int main(int argc, char const *argv[])
{
    string s;
    cin >> s;
    int n = s.length();
    char char_array[n];
    strcpy(char_array, s.c_str());
    infixTopostfix(char_array, n);

}
```

Output :-



```
a+b*c-d
abc*+d-

Intermediate Stack Values
0 +
1 *
2 -
```