Assigment-10.

Roll No. 21123 Batch-F1.

Div. SE1 Dos: Dop:

Title: Infix to Postfix Evaluation

Problem Statement- Implement C++ program for expression conversion from infix to postfix and its evaluation using stack based on given condition

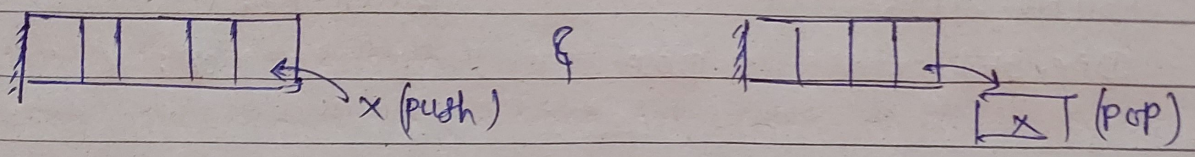
1) Operands and operator, both must be single character
2) Input postfix expression must be in desired format
3) Only '+', '-', '*', '/' are expected.

Objectives: 1) Implement a C++ program for Infix to postfix conversion of expression.
2) Implement a function to evaluate postfix expression.

Outcomes: Student will be able to
Write and excute C++ program to convert expression from infix to postfix and its evaluation.

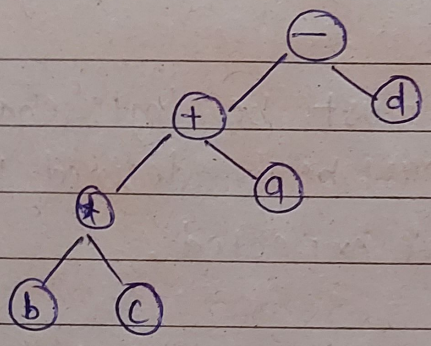
S/W & H/W Requirement-
window 10, mingw complier, Vscode, Intel-i5 9300H, 4 core, 8 logical Proccesor, 8GB RAM, 512GB, Nividia Gefore GTX 1050 GDDR5.

Theory- Stack is a data structure that follows FILO convention.
FILO stands for First In Last Out.
It has 2 main operations: push(x), pop()



Infix expression - a+b*c-d

Postfix expression - abc*+d-



Algorithm :

① Algorithm for creating stack

1. Create a class stack consist of data members int data & data[max
2. Initilize the top as -1 & create a top() fn which return the top value of the stack. // data[top].
3. ~~check~~ ~~whether the~~ Create a fn for ~~th~~checking the stack is full or Empty.
4. Create a push fn ~~returns~~ for pushing it into stack & pop for removing ~~top~~ element.

② Algorithm for Expression.

1. Create other class which is friend class of stack takes the object & two string i.e infix (taken by user) & post fix.
2. Read the expression as infix.

3. Create a function for checking prevlence(char c)
if (~~char~~ c = '*' || c = '/') return 2.
else if (c = '+' || c = '-') return 1
else return -1.

4. Algorithm for converting infix to post fix.
1. for (int i=0; i<infix.length; i++)
{
1. ~~the~~ ~~the~~ if the char is aphabet the direct take it to postfix.
2. if char is '(' push ~~the~~ into stack.
3 elif char is ')' then pop untill char is '(' and also pop '(' & then take it to postfix.
else
{ while(!sp.isEmpty() && precidence<pre(top)}
Add ~~to postfix~~ the top to postfix.
Pop the remaining operands untill stack is empty.

Pseudocode.
1. Pseudocode for creating stack.
1. int ~~def~~ top; char data[max];
2. isFull(){ if(top==max-1) return 1}
else { return 0}
3. isEmpty { if(top==-1) return 1}
else return 0

4. push(char x){ top++; data[top]=x}.
5. Pop(){ char x = data[top]; top--;}.
6. create a top1() fn which returns top value.

2. Pseudocode for Expression

1. Read the expression. & take two string as infix & postfix.
2. Checking Precidence(char c)

```
if ( c == '*' || c == '/' ) { return 2; }
else if ( c == '+' || c == '-' ) return 1 }
else return -1.
```

3. Pseudocode for Infix to postfix,

```
for (int i=0; i<Infix.length(); i++)
    if ( char is alphabet) { postfix = postfix + infix[i] }
else if ( infix[i] = '(' ) { push into stack } .
else if ( infix [i] = ')' ) { pop until
        while (sp.top() != '(' ) { postfix = postfix + sp.top()
                                   sp.pop }
    sp.pop();
else {
    while ( !sp.Empty() && prev[infix[i]] <= pre(sp.top()) )
        spostfix = postfix + sp.top()
        sp.pop()
    }
    sp.push (infix[i])

while (! sp. is Empty)
    postfix = postfix + sp.top()
    sp.pop();

cout << infix;
```