Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Information Technology

COURSE CODE: DJS22ITL302 **DATE:5/10/2023**

COURSE NAME: Data Structure Laboratory **CLASS: I1-Batch1**

NAME: Anish Sharma

Experiment No. 3

CO/LO: CO1

Aim: Implement Infix to postfix conversion and evaluation

Theory:

Infix - An infix operation is any operation of the format x op y format, such as x + y.
Postfix - An operation or expression can also be expressed as x y op, i.e. x y +, which is equivalent to writing x + y in infix. All we're trying to perform relocating the operator to the operand's right.

Program:

```
#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

#define MAX 20


char stk[20];

int top = -1;


int isEmpty()

{

   return top == -1;
```

```
}

int isFull()

{

   return top == MAX - 1;

}


char peek()

{

   return stk[top];

}


char pop()

{

   if(isEmpty())

      return -1;


   char ch = stk[top];

   top--;

   return(ch);

}
```

```
void push(char oper)
{
   if(isFull())
      printf("Stack Full!!!!");

   else{
      top++;
      stk[top] = oper;
   }
}

int checkIfOperand(char ch)
{
   return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
}

int precedence(char ch)
{
   switch (ch)
   {
   case '+':
```

```
    case '-':
        return 1;

    case '*':
    case '/':
        return 2;

    case '^':
        return 3;
    }
    return -1;
}

int covertInfixToPostfix(char* expression)
{
    int i, j;

    for (i = 0, j = -1; expression[i]; ++i)
    {
      if (checkIfOperand(expression[i]))
          expression[++j] = expression[i];
```

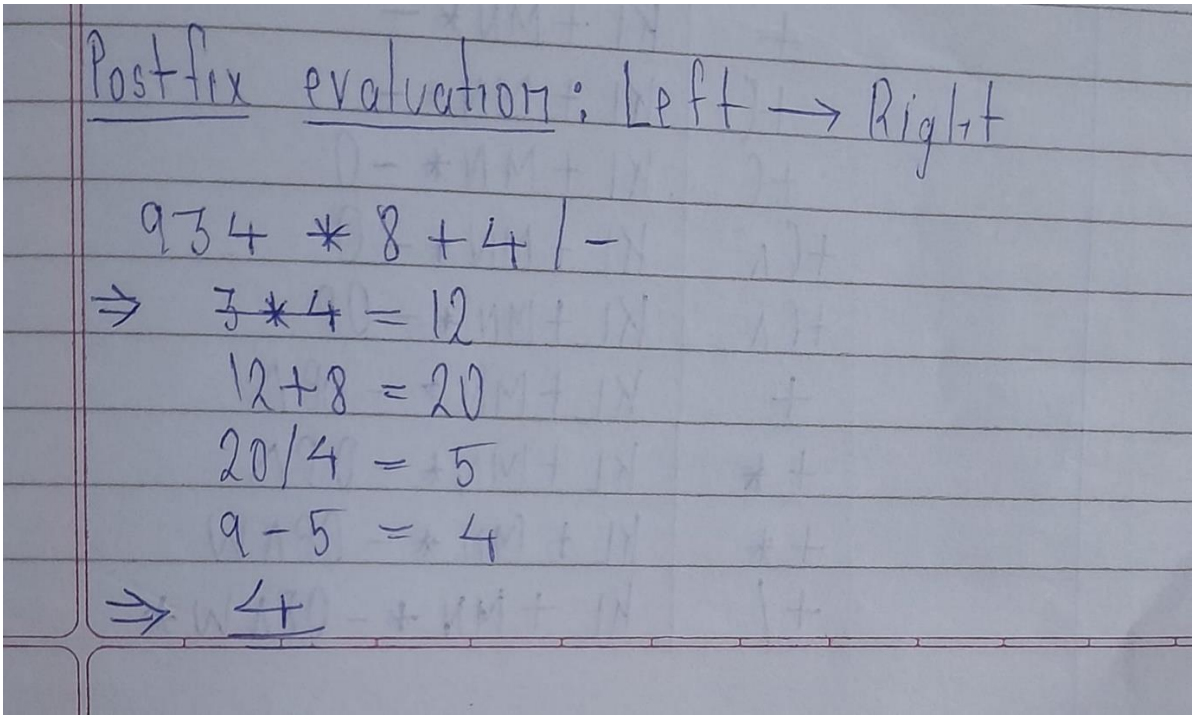
```
    else if (expression[i] == '(')
        push(expression[i]);

    else if (expression[i] == ')')
    {
        while (!isEmpty() && peek() != '(')
            expression[++j] = pop();
        if (!isEmpty() && peek() != '(')
            return -1;
        else
            pop();
    }
    else
    {
        while (!isEmpty() && precedence(expression[i]) <= precedence(peek()))
            expression[++j] = pop();
        push(expression[i]);
    }

}
```

```
    while (!isEmpty())
        expression[++j] = pop();

    expression[++j] = '\0';
    printf( "%s", expression);
}

int main()
{
char expression[] = "((x+(y*z))-w)";
    covertInfixToPostfix(expression);
    return 0;
}
```

Department of Information Technology

Output screenshots:

```
xyz*+w-

...Program finished with exit code 0
Press ENTER to exit console.
```



Conclusion:

With help of this code I learn how to convert infix to postfix and evaluation of postfix to infix.

REFERENCES:

Tutorialpoints