Data Structure and Algorithm
Lab Experiment 3
Infix to postfix and evaluation of expression

Name: Om Ashish Mishra

Registration No.: 16BCE0789

Slot: G2

The Pseudo code:

- First we use a macro to get define a variable SIZE of 50 memory space.
- Then we declare the pre-processor directive in order to write the header files.
- Then we did global declarations of the string s[SIZE] and top = -1.
- Then we did global declarations of the string c[SIZE] and top1 = -1.
- Then the function push is made to insert the elements into stack if not full.
- Then the function pop is made to remove the elements from stack if not empty.(The parameter is a character)
- Then the function insert is made to insert the elements into stack if not full.
- Then the function del is made to remove the elements from stack if not empty.(The parameter is an integer)
- Then the function precedence is called to check the order of precedency that is first # which check whether the stack has approached to its last element, then comes '(' , then comes '*' and '/' and at last '-' and '+'.
- Then in the main function we take the input string and then we check each character of the string.
- First we check ')' , if it's a bracket, we pop the elements out of it till '(', else we add the elements to the stack.
- Then after the postfix is obtained from the infix. We go for evaluation.
- In this case we check if it contains number and if so we proceed.

- Then we pop to numbers if an operator is found and put the final result back into the stack
- At the end there ill we only 1 element in the stack and it will we the answer to the evaluation.

The Code:

```
#define SIZE 50          /* Size of Stack */

#include <ctype.h>

char s[SIZE];

int top=-1;      /* Global declarations */

char c[SIZE];

int top1=-1;

void push(char elem)

{                    /* Function for PUSH operation */

   s[++top]=elem;

}


char pop()

{                   /* Function for POP operation */

   return(s[top--]);

}


insert(int elem)

{                    /* Function for PUSH operation */

 c[++top1]=elem;

}

int del()

{
```

```
    return(c[top1--]);
}
int pr(char elem)
{               /* Function for precedence */
    switch(elem)
    {
    case '#': return 0;
    case '(': return 1;
    case '+':
    case '-': return 2;
    case '*':
    case '/': return 3;
    }
}

void main()
{                       /* Main Program */
    char infx[50],pofx[50],ch,elem;
    int i=0,k=0;
    int op2,op1;
    printf("\n\nRead the Infix Expression ? ");
    scanf("%s",&infx);
    push('#');
    while( (ch=infx[i++]) != '\0')
    {
```

```
    if( ch == '(') push(ch);
    else
        if(isalnum(ch)) pofx[k++]=ch;
        else
            if( ch == ')')
            {
                while( s[top] != '(')
                    pofx[k++]=pop();
                elem=pop(); /* Remove ( */
            }
            else
            {      /* Operator */
                while( pr(s[top]) >= pr(ch) )
                    pofx[k++]=pop();
                push(ch);
            }
}
while( s[top] != '#')     /* Pop from stack till empty */
    pofx[k++]=pop();
pofx[k]='\0';         /* Make pofx as valid string */
printf("\n\nGiven Infix Expn: %s  Postfix Expn: %s\n",infx,pofx);



char post[50];
i=0;
```

```
printf("\n\n the Postfix Expression ? ");
scanf("%s",&post);
while( (ch=post[i++]) != '\0')
{
   if(isdigit(ch))
      insert(ch-'0'); /* Push the operand */
   else
   {      /* Operator,pop two  operands */
      op2=del();
      op1=del();
      switch(ch)
      {
         case '+':
            insert(op1+op2);
            break;
         case '-':
            insert(op1-op2);
            break;
         case '*':
            insert(op1*op2);
            break;
         case '/':
            insert(op1/op2);
            break;
      }
```

```
        }
    }
    printf("\n Given Postfix Expn: %s\n",post);
    printf("\n Result after Evaluation: %d\n",s[top1]);
}
```

The Output

```
C:\Users\OM\(OM)\Untitled2.exe

Read the Infix Expression ? 7*5

Given Infix Expn: 7*5  Postfix Expn: 75*

 the Postfix Expression ? 75*

 Given Postfix Expn: 75*

 Result after Evaluation: 35

Process returned 30 (0x1E)   execution time : 18.063 s
Press any key to continue.
```