Name: Saloni Vishwakarma

Roll no: 13 (C1-Batch)

PRACTICAL NO: 2

Aim: To study and implement Stack ADT and write a function to convert infix expression to postfix expression and evaluate the postfix expression using a Stack.

Code for Stack operations:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 1
int st[MAX], top=-1;
void push (int st[], int val);
int pop (int st[]);
int peek (int st[]);
void display (int st[]);
int main (int argc, char *argv[]) {
int val, option;
do{
printf("\n *****MAIN MENU*****");
printf("\n 1. PUSH");
printf("\n 2. POP");
printf("\n 3. PEEK");
printf("\n 4. DISPLAY");
printf("\n 5. EXIT");
printf("\n Enter your option: ");
```

```
scanf("%d", &option);

switch (option){
case 1:
printf("\n Enter the number to be pushed on stack: ");
scanf("%d", &val);
push(st, val);
break;
case 2:
val = pop(st);
if(val != -1)
printf("\n The value deleted from stack is: %d", val);
break;
case 3:
val = peek(st);
if(val != -1)
printf("\n The value stored at top of stack is: %d", val);
break;
case 4:
display(st);
break;
} }while(option != 5);
return 0; }

void push (int st[], int val){
if(top == MAX-1){
printf("\n STACK OVERFLOW. NO INSERTION.");
}
else{
top++;
```

```
st[top] = val;
printf("Successful insertion");
} }

int pop (int st[]) {
int val;
if(top == -1) {
printf("\n STACK UNDERFLOW. NO DELETION.");
return -1;
}
else {
val = st[top];
top--;
printf("Successful deletion");
return val;
} }

void display (int st[]){
int i;
if(top == -1)
printf("\n STACK IS EMPTY");
else {
for(i=top;i>=0;i--)
printf("\n %d",st[i]);
printf("\n"); // Added for formatting purposes
} }

int peek (int st[]) {
if(top == -1) {
printf("\n STACK IS EMPTY");
```

return -1;

}

else

return (st[top]);

}

TEST CASES:

1. Push: Insert an element in a Stack.

a. Successful Insertion.

b. Stack Overflow. No insertion.

```
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the number to be pushed on stack: 5
Successful insertion
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the number to be pushed on stack: 6

STACK OVERFLOW. NO INSERTION.
```

2. Pop: Delete an element in a Stack.

a. Successful Deletion.

b. Stack Underflow. No Deletion.

```
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 2
Successful deletion
The value deleted from stack is: 5
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 2

STACK UNDERFLOW. NO DELETION.
```

3. Peek: Visit and print the topmost element of the stack.

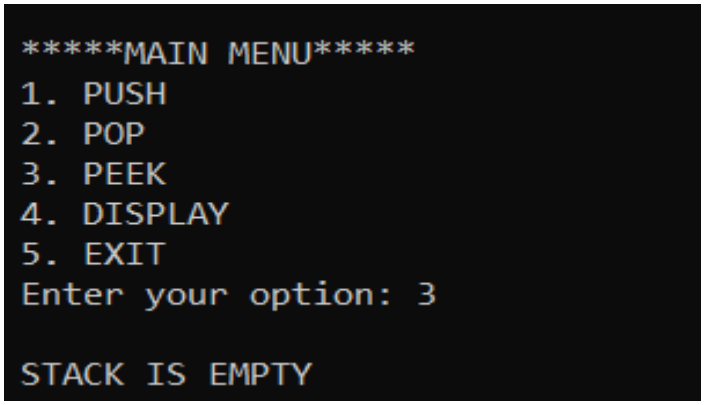
a. Successful Operation. Print the topmost element.

b. Stack Underflow. Print “Stack is Empty”.

```
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the number to be pushed on stack: 9
Successful insertion
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 3

The value stored at top of stack is: 9
```



Code for conversion of infix to postfix expression:

```
#include <stdio.h>
#include <ctype.h>
void push(char x);
char pop();
int priority(char x);

struct stack
{
   char arr[100];
   int top;
} s1;
int main()
{
   s1.top = -1;
   char exp[100];
   char *e, x;
   printf("Enter the expression : ");
   scanf("%s", exp);
   printf("\n");
   e = exp;
   while (*e != '\0')
```

```
    {
        if (isalnum(*e))
            printf("%c ", *e);
        else if (*e == '(')
            push(*e);
        else if (*e == ')')
        {
            while ((x = pop()) != '(')
                printf("%c ", x);
        }
        else
        {
            while (priority(s1.arr[s1.top]) >= priority(*e))
                printf("%c ", pop());
            push(*e);
        }
        e++;
    }

    while (s1.top != -1)
    {
        printf("%c ", pop());
    }
    return 0;
}

void push (char x)
{
    s1.arr[++s1.top] = x;
}
```

```
char pop ()
{
   if (s1.top == -1)
      return -1;
   else
      return s1.arr[s1.top--];
}
int priority (char x)
{
   if (x == '(')
      return 0;
   if (x == '+' || x == '-')
      return 1;
   if (x == '*' || x == '/')
      return 2;
   return 0;
}
```

OUTPUT:

```
Enter any infix expression : A*B

The corresponding postfix expression is : AB*
```

Code for evaluation of postfix expressions:

```
#include<stdio.h>
int stack[20];
int top = -1;

void push(int x)
```

```
{
   stack[++top] = x;
}

int pop()
{
   return stack[top--];
}

int main()
{
   char exp[20];
   char *e;
   int n1,n2,n3,num;
   printf("Enter the expression :: ");
   scanf("%s",exp);
   e = exp;
   while(*e != '\0')
   {
      if(isdigit(*e))
      {
         num = *e - 48;
         push(num);
      }
      else
      {
         n1 = pop();
         n2 = pop();
         switch(*e)
         {
```

```
            case '+':
            {
               n3 = n1 + n2;
               break;
            }
            case '-':
            {
               n3 = n2 - n1;
               break;
            }
            case '*':
            {
               n3 = n1 * n2;
               break;
            }
            case '/':
            {
               n3 = n2 / n1;
               break;
            }
            }
            push(n3);
         }
         e++;
      }
      printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
      return 0;
   }
```

OUTPUT:

```
Enter the expression :: 45+

The result of expression 45+  =  9



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

TEST CASES:

4. Execute Conversion of Infix to Postfix and Evaluation of Postfix Expressions using the following examples:

a)4+5*6

```
Enter the expression : 4+5*6

4 5 6 * +

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

b) 3555/*+22*+

```
Enter the expression :: 3555/*+22+

The result of expression 3555/*+22+  =  4



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