

Laboratory 5

1. Questions

1. Implement the PUSH, POP and PRINT operations on stack
2. Write a C program to convert infix to postfix notation using stack
3. Write a C program to evaluate a postfix expression.

2. Algorithm

1. Algorithm to Implement the PUSH, POP and PRINT operations on stack

Step 1= start

Step 2. Ask the user for the operation like push, pop, display and exit. Use the variable top to represent the top of the stack.

2. According to the option entered, access its respective function using switch statement.

3. In the function push(), firstly check if the stack is full. If it is, then print the output as "Stack is Full". Otherwise take the number to be inserted as input and store it in the variable num. Copy the number to the array stk[] and increment the variable top by 1.

4. In the function pop(), firstly check if the stack is empty. If it is, then print the output as "Stack is Empty". Otherwise print the top most element of the array stk[] and decrement the variable top by 1.

5. In the function display(), using for loop print all the elements of the array.

6. Exit.

2. Algorithm to convert infix to postfix notation using stack

1. Scan the infix expression from left to right.

2. If the scanned character is an operand, output it.

3. Else,

.....3.1 If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty or the stack

contains a '(', push it.

.....3.2 Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)

4. If the scanned character is an '(', push it to the stack.

5. If the scanned character is an ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.

6. Repeat steps 2-6 until infix expression is scanned.

7. Print the output

8. Pop and output from the stack until it is not empty.

3. Algorithm to evaluate a postfix expression.

1) Create a stack to store operands (or values).

2) Scan the given expression and do following for every scanned element.

.....a) If the element is a number, push it into the stack

.....b) If the element is a operator, pop operands for the operator from stack.

Evaluate the operator and push the result back to the stack

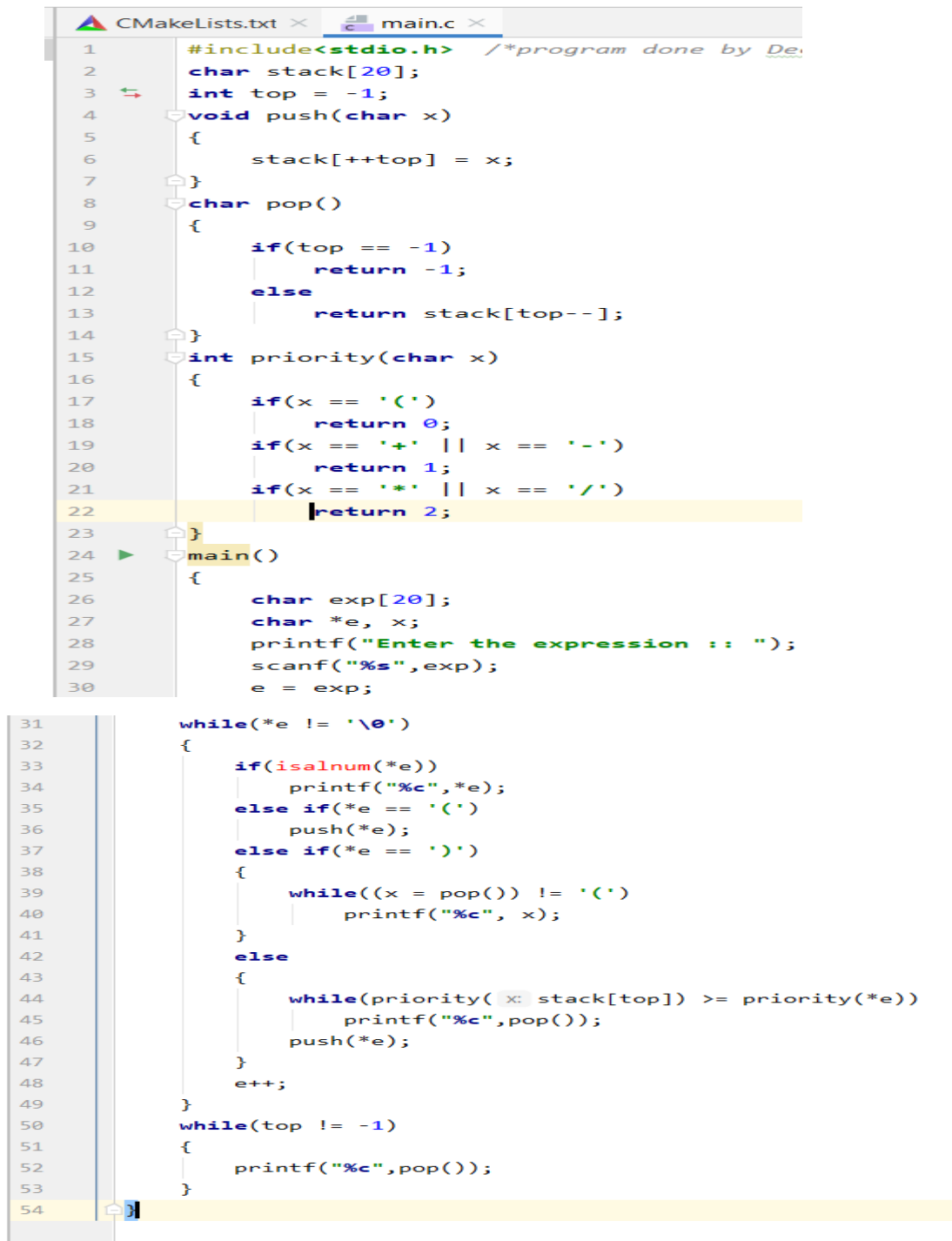
3) When the expression is ended, the number in the stack is the final answer

3. Program

```
1
2  * C program to implement stack. Stack is a LIFO data structure.
3  * Stack operations: PUSH(insert operation), POP(Delete operation)
4  * and Display stack. program done by Deepak R
5  */
6  #include <stdio.h>
7  #define MAXSIZE 5
8
9  struct stack
10 {
11     int stk[MAXSIZE];
12     int top;
13 };
14 typedef struct stack STACK;
15 STACK s;
16
17 void push(void);
18 int pop(void);
19 void display(void);
20
21 void main ()
22 {
23     int choice;
24     int option = 1;
25     s.top = -1;
26
27     printf ("STACK OPERATION\n");
28     while (option)
29     {
30         printf ("-----\n");
31         printf ("      1  -->  PUSH           \n");
32         printf ("      2  -->  POP            \n");
33         printf ("      3  -->  DISPLAY        \n");
34         printf ("      4  -->  EXIT           \n");
35         printf ("-----\n");
36
37         printf ("Enter your choice\n");
38         scanf ("%d", &choice);
39         switch (choice)
40         {
41             case 1:
42                 push();
43                 break;
44             case 2:
45                 pop();
46                 break;
47             case 3:
48                 display();
49                 break;
50             case 4:
51                 return;
52             }
53         fflush (stdin);
54         printf ("Do you want to continue(Type 0 or 1)?\n");
55         scanf ("%d", &option);
56     }
57 }
```

```
58  /* Function to add an element to the stack */
59  void push ()
60  {
61      int num;
62      if (s.top == (MAXSIZE - 1))
63      {
64          printf ("Stack is Full\n");
65          return;
66      }
67      else
68      {
69          printf ("Enter the element to be pushed\n");
70          scanf ("%d", &num);
71          s.top = s.top + 1;
72          s.stk[s.top] = num;
73      }
74      return;
75  }
76  /* Function to delete an element from the stack */
77  int pop ()
78  {
79      int num;
80      if (s.top == -1)
81      {
82          printf ("Stack is Empty\n");
83          return (s.top);
84      }
85      else
86      {
87          num = s.stk[s.top];
88          printf ("popped element is = %dn", s.stk[s.top]);
89          s.top = s.top - 1;
90      }
91      return(num);
92  }
93  /* Function to display the status of the stack */
94  void display ()
95  {
96      int i;
97      if (s.top == -1)
98      {
99          printf ("Stack is empty\n");
100         return;
101     }
102     else
103     {
104         printf ("\n The status of the stack is \n");
105         for (i = s.top; i >= 0; i--)
106         {
107             printf ("%d\n", s.stk[i]);
108         }
109     }
110     printf ("\n");
}
```

Fig 1 program to Implement the PUSH, POP and PRINT operations on stack



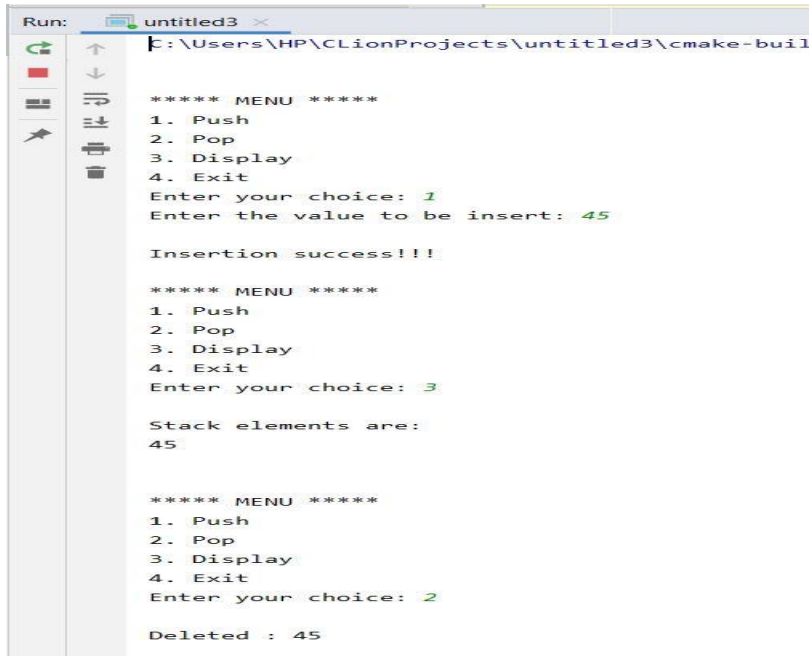
```
1  #include<stdio.h> /*program done by De
2  char stack[20];
3  int top = -1;
4  void push(char x)
5  {
6      stack[++top] = x;
7  }
8  char pop()
9  {
10     if(top == -1)
11         return -1;
12     else
13         return stack[top--];
14 }
15 int priority(char x)
16 {
17     if(x == '(')
18         return 0;
19     if(x == '+' || x == '-')
20         return 1;
21     if(x == '*' || x == '/')
22         return 2;
23 }
24 main()
25 {
26     char exp[20];
27     char *e, x;
28     printf("Enter the expression :: ");
29     scanf("%s",exp);
30     e = exp;
31
32     while(*e != '\0')
33     {
34         if(isalnum(*e))
35             printf("%c",*e);
36         else if(*e == '(')
37             push(*e);
38         else if(*e == ')')
39         {
40             while((x = pop()) != '(')
41                 printf("%c", x);
42         }
43         else
44         {
45             while(priority(stack[top]) >= priority(*e))
46                 printf("%c",pop());
47             push(*e);
48         }
49         e++;
50     }
51     while(top != -1)
52     {
53         printf("%c",pop());
54     }
```

Fig 2 program convert infix to postfix notation using stack

```
CMakeLists.txt x main.c x
1  #include<stdio.h>
2  int stack[20];
3  int top = -1;
4
5  void push(int x)
6  {
7      stack[++top] = x;
8  }
9  int pop()
10 {
11     return stack[top--];
12 }
13 int main()
14 {
15     char exp[20];
16     char *e;
17     int n1,n2,n3,num;
18     printf("Enter the expression :: ");
19     scanf("%s",exp);
20     e = exp;
21     while(*e != '\0')
22     {
23         if(isdigit(*e))
24         {
25             num = *e - 48;
26             push(num);
27         }
28         else
29         {
30             n1 = pop();
31             n2 = pop();
32
33             switch(*e)
34             {
35                 case '+':
36                 {
37                     n3 = n1 + n2;
38                     break;
39                 }
40                 case '-':
41                 {
42                     n3 = n2 - n1;
43                     break;
44                 }
45                 case '*':
46                 {
47                     n3 = n1 * n2;
48                     break;
49                 }
50                 case '/':
51                 {
52                     n3 = n2 / n1;
53                     break;
54                 }
55             }
56             push(n3);
57             e++;
58         }
59     }
60     printf("\nThe result of expression %s = %d\n\n",exp,pop());
61     return 0;
62 }
```

Fig 3 program to evaluate a postfix expression.

4. Presentation of Results



```
Run: untitled3 x
E:\Users\HP\CLionProjects\untitled3\cmake-build

***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 45

Insertion success!!!

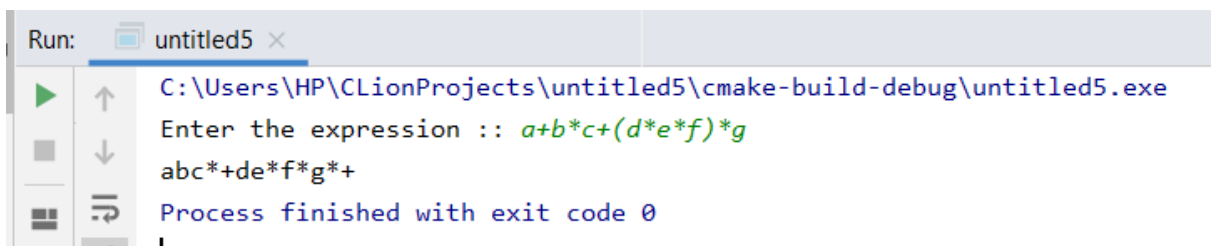
***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3

Stack elements are:
45

***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2

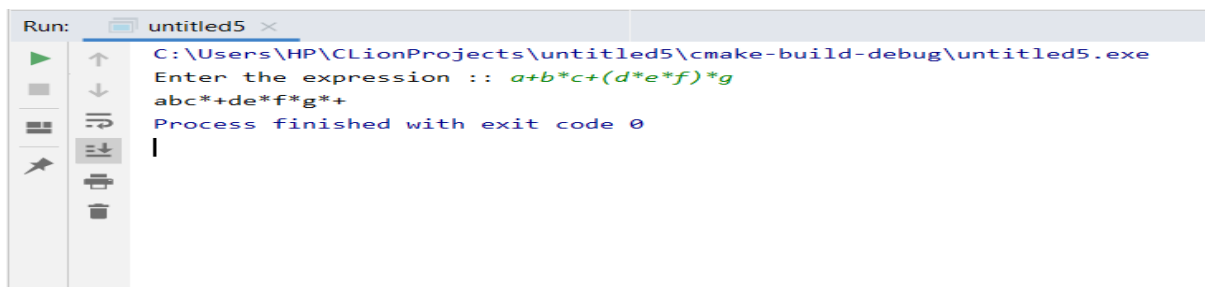
Deleted : 45
```

Fig 4 Result of program to Implement the PUSH, POP and PRINT operations on stack



```
Run: untitled5 x
C:\Users\HP\CLionProjects\untitled5\cmake-build-debug\untitled5.exe
Enter the expression :: a+b*c+(d*e*f)*g
abc*+de*f*g*+
Process finished with exit code 0
```

Fig 5 Result of program convert infix to postfix notation using stack



```
Run: untitled5 x
C:\Users\HP\CLionProjects\untitled5\cmake-build-debug\untitled5.exe
Enter the expression :: a+b*c+(d*e*f)*g
abc*+de*f*g*+
Process finished with exit code 0
```

Fig 6 Result of program to evaluate a postfix expression

5. Conclusions

In this lab we learnt how to Implement the PUSH, POP and PRINT operations on stack and to Write a C program to convert infix to postfix notation using stack at last we learnt to Write a C program to evaluate a postfix expression.