1/1/24

Lab - 2 :

1) Write a program to simulate the working of stack using an array with the following :
a) Push
b) Pop
c) Display
The program should print appropriate messages for stack overflow, stack underflow

ans)
```
# include <stdio.h>
# define SIZE 10
int top = -1, stack [SIZE];
void push ()
{
    int x ;
    if (top == SIZE-1)
    {
        printf ("\n overflow");
    }
    else
    {
        printf ("\n Enter the element to be
        added :");
        scanf ("%d", &x);
        top = top +1;
        stack [top] = x ;
    }
}
void pop ()
{
    if (top == -1)
    {
```

Executed

[illegible] 11/1/24

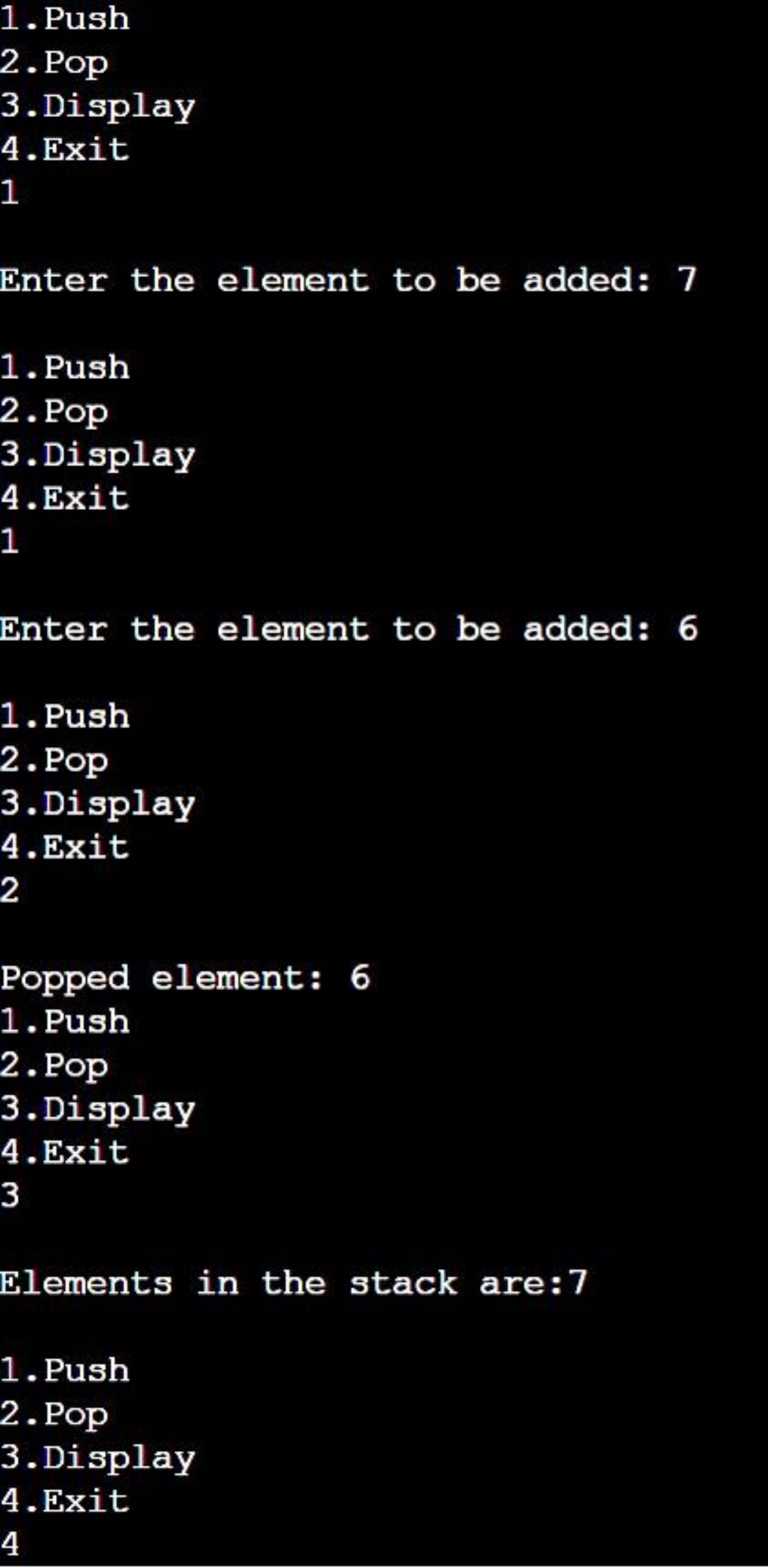
```
            printf("\n underflow");
        }
        else
        {
            printf("\n Popped element: %d, stack[top]
            top = top -1;
        }
}
void display()
{
        printf(" The elements in the stack are"
        for (int i = 0; i <= top; i++)
        {
            printf("%d\n", stack[i]);
        }
}
int main ()
{
    int choice;
    while (1)
    {
        printf("1. Push \n 2. Pop \n 3. Display \n
        4. Exit");
        scanf("%d", & choice);
        switch (choice)
        {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf(" Invalid choice");
        }
    }
return 0;
}
```

<u>OUTPUT</u>:

```
1. Push
2. Pop
3. Display
4. Exit
1

Enter the element to be added: 7
1. Push
2. Pop
3. Display
4. Exit
3

The elements in the stack are:
7
```



```
1.Push
2.Pop
3.Display
4.Exit
1

Enter the element to be added: 7

1.Push
2.Pop
3.Display
4.Exit
1

Enter the element to be added: 6

1.Push
2.Pop
3.Display
4.Exit
2

Popped element: 6
1.Push
2.Pop
3.Display
4.Exit
3

Elements in the stack are:7

1.Push
2.Pop
3.Display
4.Exit
4
```

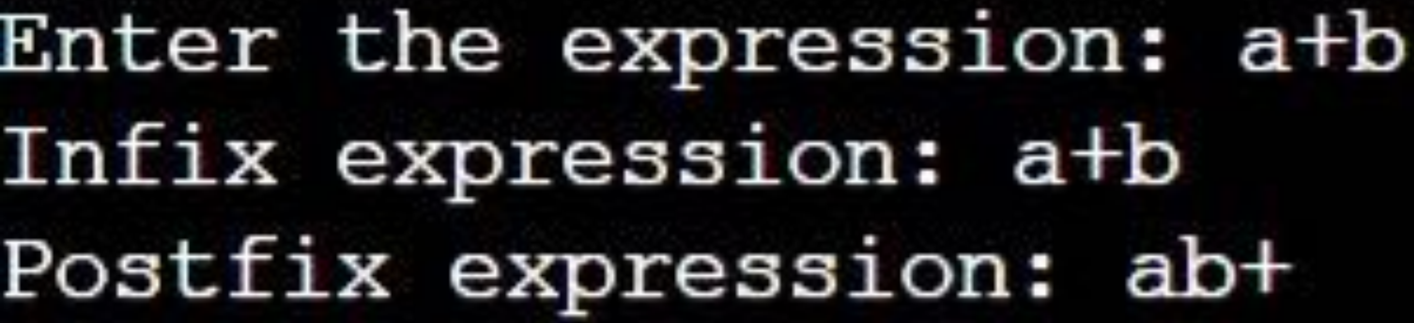
2) Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

ans)
```
#include <stdio.h>
#include <string.h>
int index1=0, pos=0, top=-1, length;
char symbol, temp, infix[50], postfix[50], stack[50]
void infixtopostfix();
void push (char symbol);
char pop();
int precedence(char symbol);
void main()
{
        printf("Enter the expression :");
        scanf("%s", infix);
        infixtopostfix();
        printf("infix expression : %s", infix);
        printf("\n Postfix expression : %s", postfix);
}
void infixtopostfix()
{
     length = strlen(infix);
     while (index1 < length)
     {
          symbol= infix[index1];
          switch( symbol)
          {
               case '(': push (symbol);
                         break;
case ')' :  temp = pop();
            while (temp != '(')
            {
                postfix [pos] = temp;
                pos ++;
                temp = pop();
            }
            break;
case '+' :
  case '-':
  case '*':
  case '/':
      while (precedence (stack [top]) >=
                           precedence (symbol))
      {
          temp = pop();
          postfix [ pos++] = temp;
        }
        push (symbol);
        break;
   default :  postfix [ pos++] = symbol;
 }
 index1++;
 }
  while (top>0)
  {
     temp = pop();
     postfix [pos++] = temp;
   }
}
void  push  (char  symbol)
{
   top = top + 1;
   stack [top] = symbol;
}
```

```
char pop()
{
    char s;
    s = stack [top];
    top = top -1;
    return (s);
}
int precedence (char symbol)
{
    int p;
    switch (symbol)
    {
        case '*':
        case '/':  p=2;
                   break;
        case '+':
        case '-':  p=1;
                   break;
}
return (p);
}
```

OUTPUT:

```
Enter the expression: a+b
Infix expression: a+b
Postfix expression: ab+
```



```
Enter the expression: a+b
Infix expression: a+b
Postfix expression: ab+
```