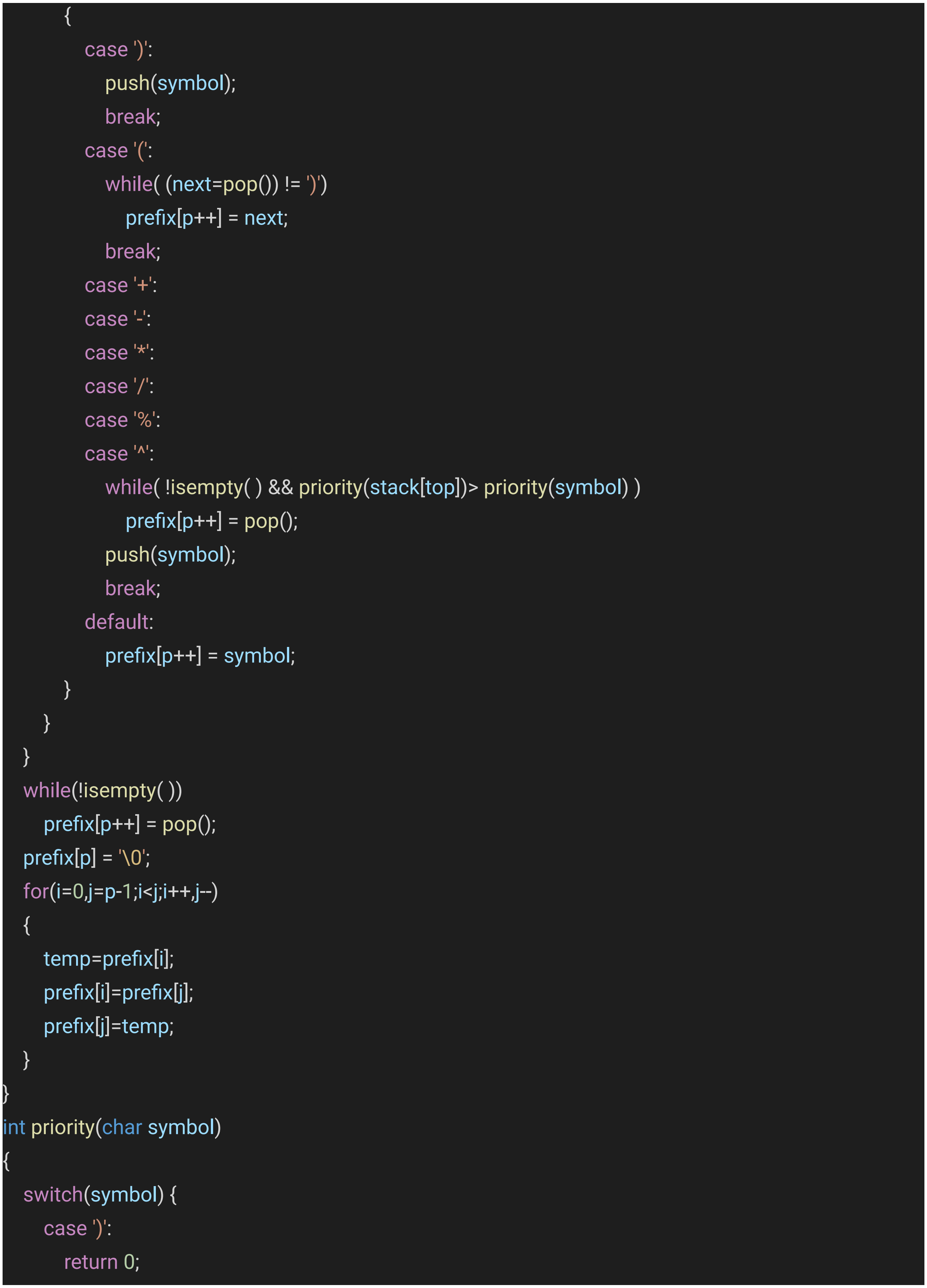
Assignment- 1

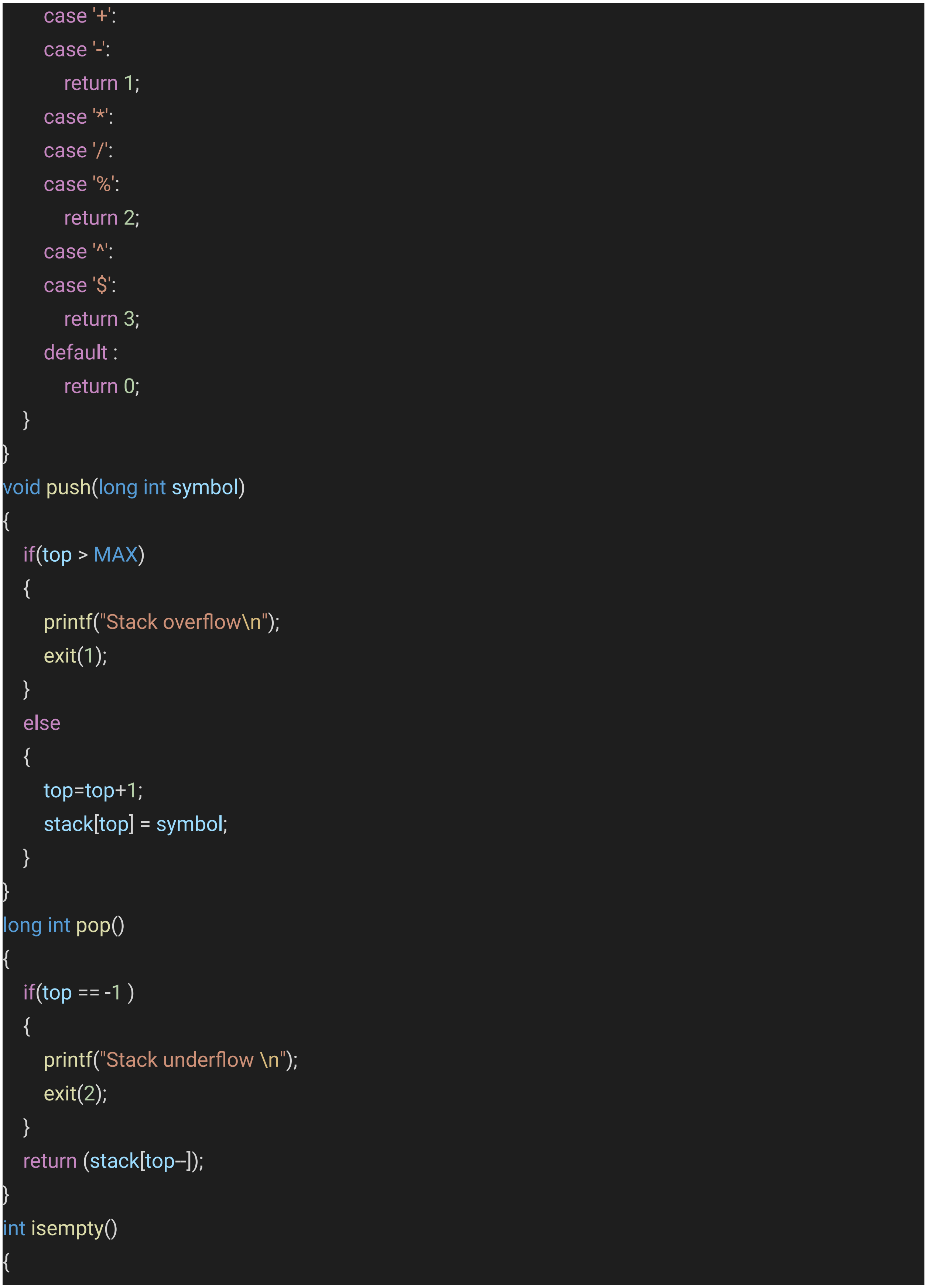
1)Develop a menu driven program to convert the given infix expression to postfix and prefix form.

Program

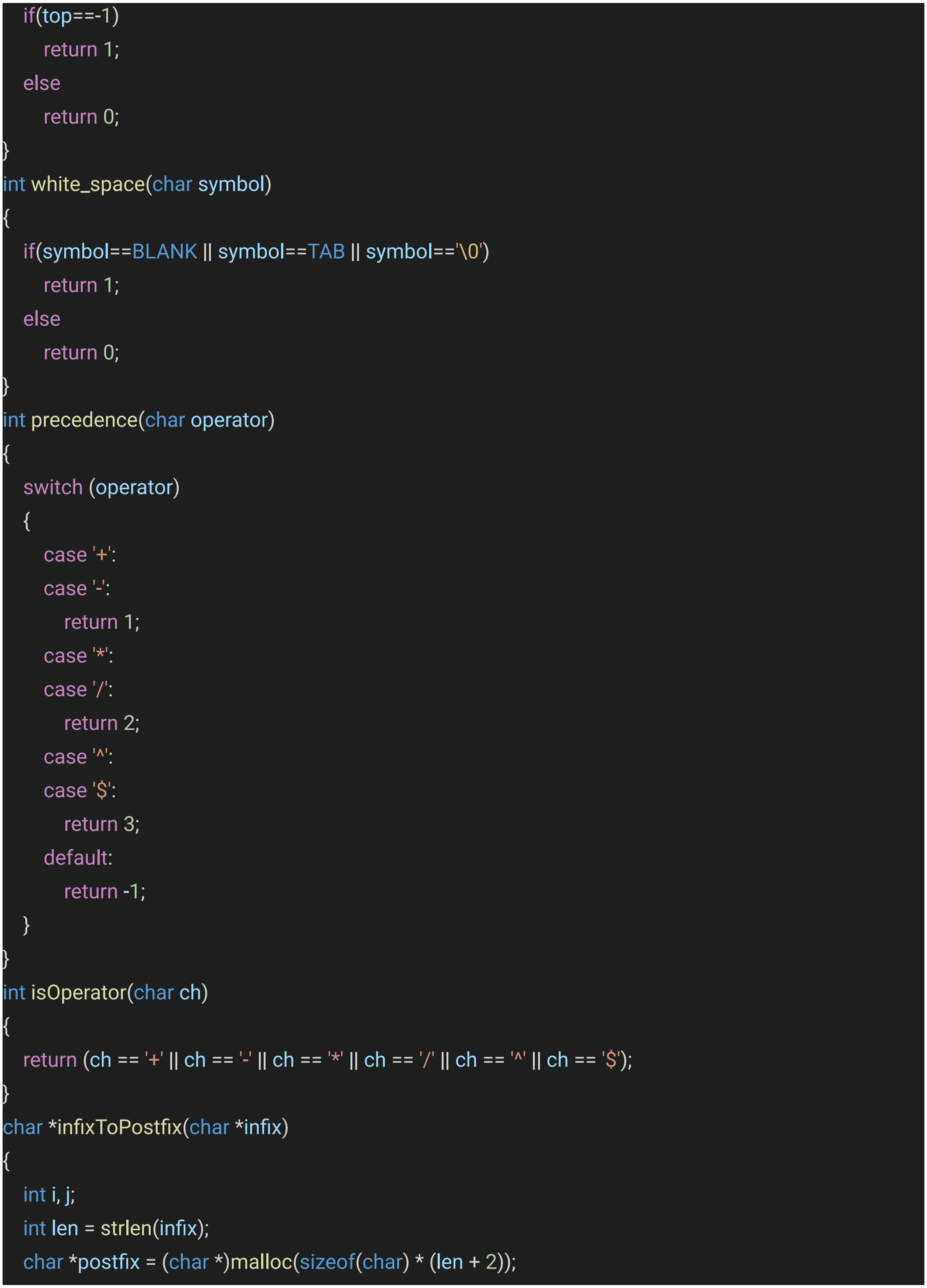
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include<ctype.h>
#include<math.h>
#define MAX_EXPR_SIZE 100
#define BLANK ' '
#define TAB '\t'
#define MAX 50
long int pop();
char infx[MAX],prefix[MAX];
long int stack[MAX];
int top;
int isempty();
int white_space(char symbol);
void infix_to_prefix();
int priority(char symbol);
void push(long int symbol);
void infix_to_prefix()
{
    int i,j,p,n;
    char next ;
    char symbol;
    char temp;
    n=strlen(infx);
    p=0;
    for(i=n-1; i>=0; i–)
    {
        symbol=infx[i];
        if(!white_space(symbol))
        {
            switch(symbol)
```



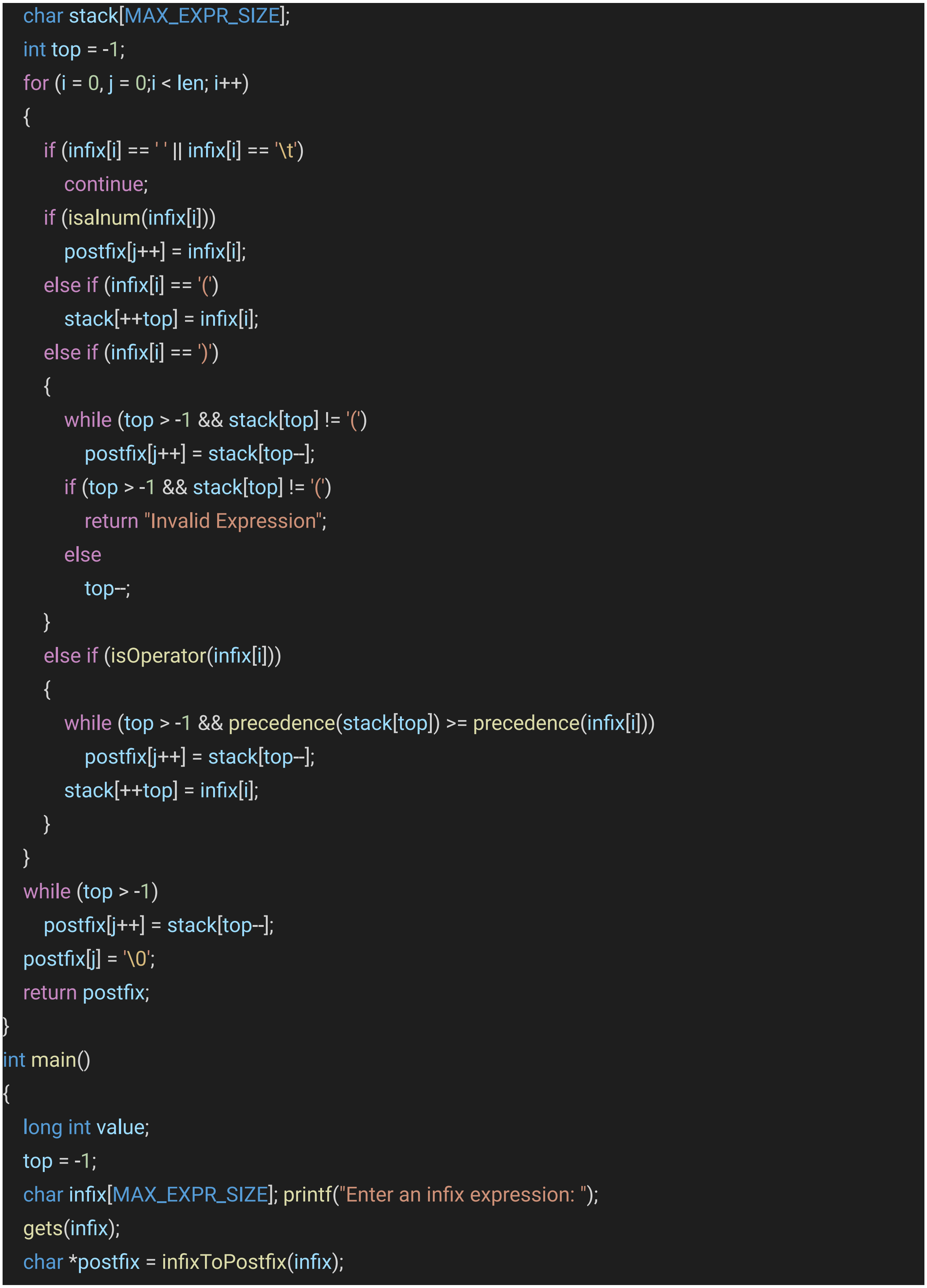
```
            {
                case ')':
                    push(symbol);
                    break;
                case '(':
                    while( (next=pop()) != ')')
                        prefix[p++] = next;
                    break;
                case '+':
                case '-':
                case '*':
                case '/':
                case '%':
                case '^':
                    while( !isempty( ) && priority(stack[top])> priority(symbol) )
                        prefix[p++] = pop();
                    push(symbol);
                    break;
                default:
                    prefix[p++] = symbol;
            }
        }
    }
    while(!isempty( ))
        prefix[p++] = pop();
    prefix[p] = '\0';
    for(i=0,j=p-1;i<j;i++,j--)
    {
        temp=prefix[i];
        prefix[i]=prefix[j];
        prefix[j]=temp;
    }
}
int priority(char symbol)
{
    switch(symbol) {
        case ')':
            return 0;
```



```
    case '+':
    case '-':
      return 1;
    case '*':
    case '/':
    case '%':
      return 2;
    case '^':
    case '$':
      return 3;
    default :
      return 0;
  }
}
void push(long int symbol)
{
  if(top > MAX)
  {
    printf("Stack overflow\n");
    exit(1);
  }
  else
  {
    top=top+1;
    stack[top] = symbol;
  }
}
long int pop()
{
  if(top == -1 )
  {
    printf("Stack underflow \n");
    exit(2);
  }
  return (stack[top–]);
}
int isempty()
{
```



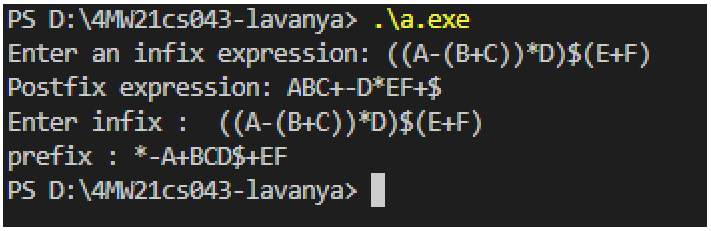
```
    if(top==-1)
        return 1;
    else
        return 0;
}
int white_space(char symbol)
{
    if(symbol==BLANK || symbol==TAB || symbol=='\0')
        return 1;
    else
        return 0;
}
int precedence(char operator)
{
    switch (operator)
    {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
        case '$':
            return 3;
        default:
            return -1;
    }
}
int isOperator(char ch)
{
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^' || ch == '$');
}
char *infixToPostfix(char *infix)
{
    int i, j;
    int len = strlen(infix);
    char *postfix = (char *)malloc(sizeof(char) * (len + 2));
```



```
    char stack[MAX_EXPR_SIZE];
    int top = -1;
    for (i = 0, j = 0;i < len; i++)
    {
        if (infix[i] == ' ' || infix[i] == '\t')
            continue;
        if (isalnum(infix[i]))
            postfix[j++] = infix[i];
        else if (infix[i] == '(')
            stack[++top] = infix[i];
        else if (infix[i] == ')')
        {
            while (top > -1 && stack[top] != '(')
                postfix[j++] = stack[top–];
            if (top > -1 && stack[top] != '(')
                return "Invalid Expression";
            else
                top–;
        }
        else if (isOperator(infix[i]))
        {
            while (top > -1 && precedence(stack[top]) >= precedence(infix[i]))
                postfix[j++] = stack[top–];
            stack[++top] = infix[i];
        }
    }
    while (top > -1)
        postfix[j++] = stack[top–];
    postfix[j] = '\0';
    return postfix;
}
int main()
{
    long int value;
    top = -1;
    char infix[MAX_EXPR_SIZE]; printf("Enter an infix expression: ");
    gets(infix);
    char *postfix = infixToPostfix(infix);
```

```c
  printf("Postfix expression: %s\n", postfix);
  free(postfix);
  printf("Enter infix : ");
  gets(infx);
  infix_to_prefix();
  printf("prefix : %s\n",prefix);
  return 0;
}
```

Output

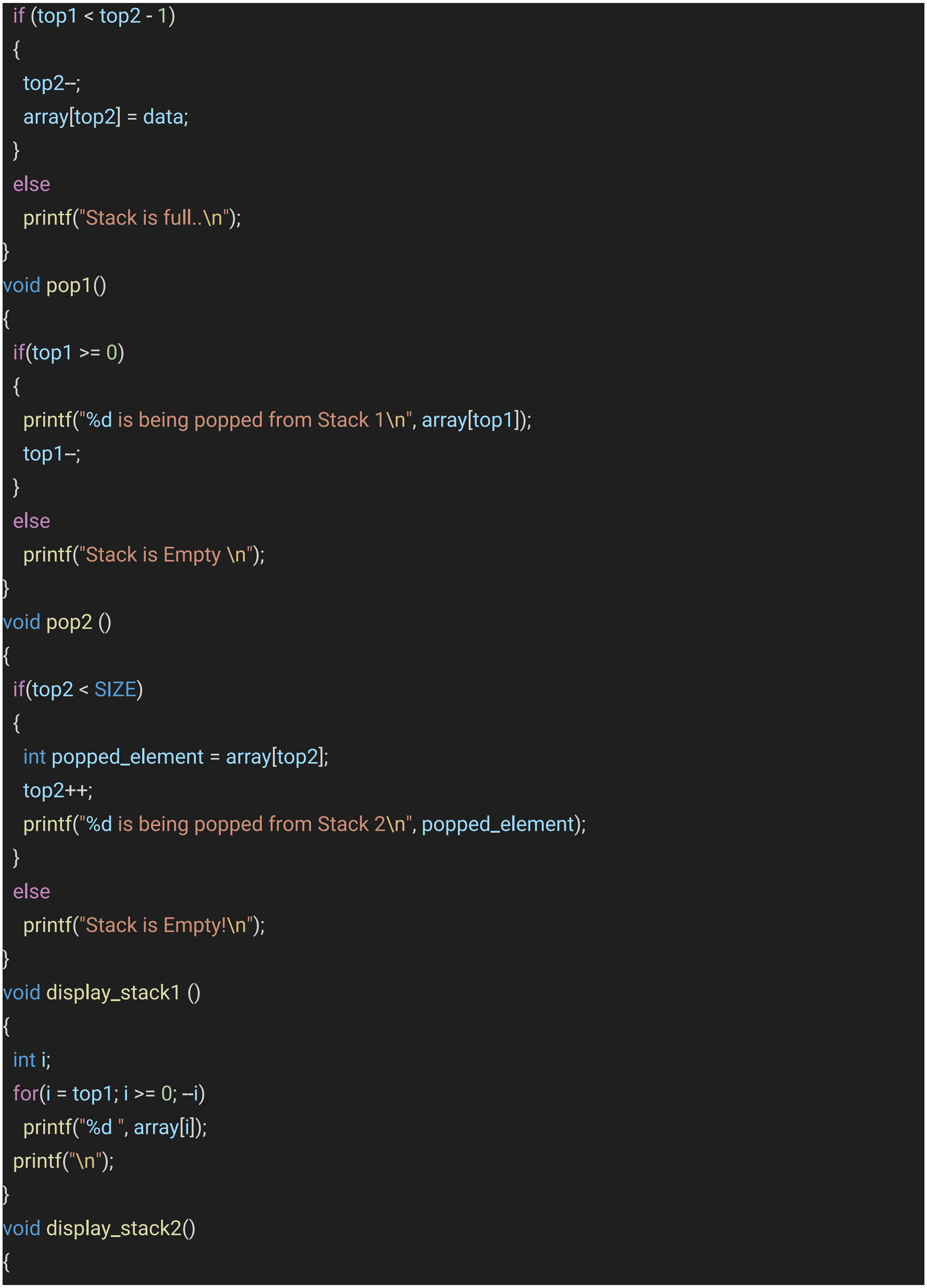


```
PS D:\4MW21cs043-lavanya> .\a.exe
Enter an infix expression: ((A-(B+C))*D)$(E+F)
Postfix expression: ABC+-D*EF+$
Enter infix :  ((A-(B+C))*D)$(E+F)
prefix : *-A+BCD$+EF
PS D:\4MW21cs043-lavanya>
```

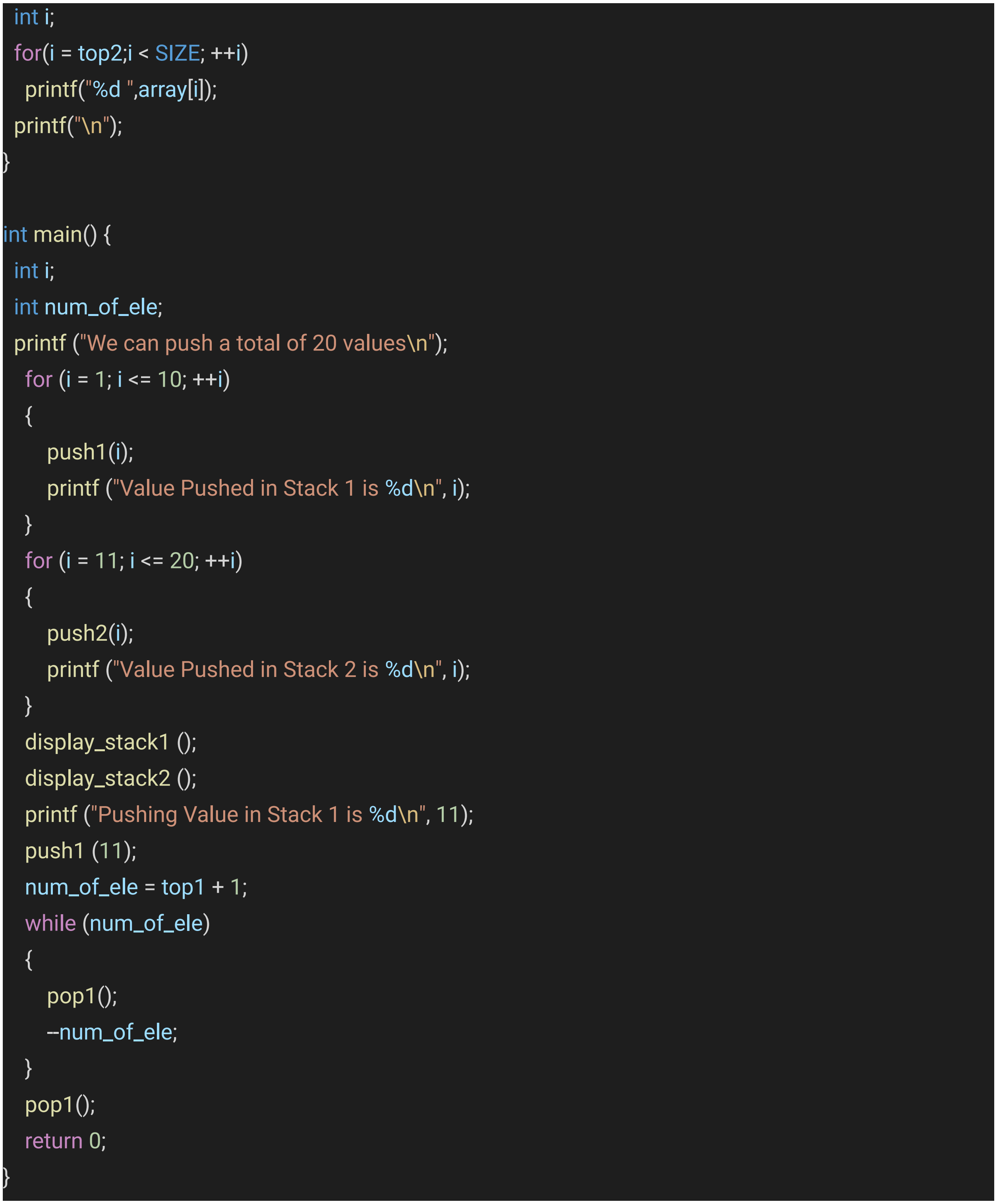
2) Implement multiple stacks using a one dimensional array.

Program

```c
#include <stdio.h>
#define SIZE 20
int array[SIZE];
int top1= -1;
int top2 = SIZE;
void push1(int data)
{
 if (top1<top2-1)
 {
  top1++;
  array[top1] = data;
 }
 else
  printf("Stack is full\n");
}
void push2(int data)
{
```

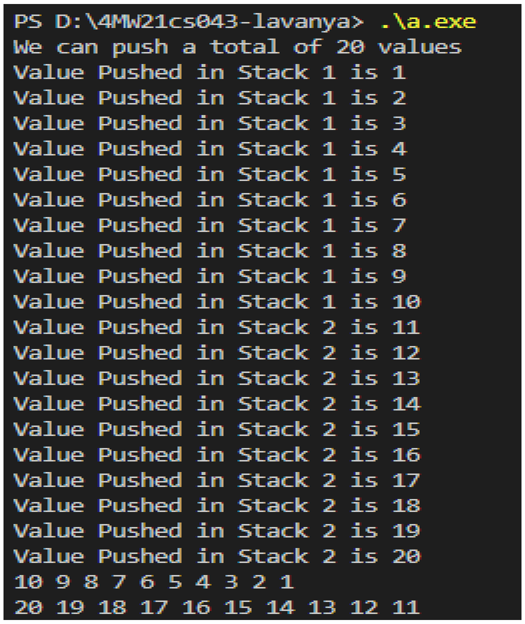


```
 if (top1 < top2 - 1)
 {
  top2–;
  array[top2] = data;
 }
 else
  printf("Stack is full..\n");
}
void pop1()
{
 if(top1 >= 0)
 {
  printf("%d is being popped from Stack 1\n", array[top1]);
  top1–;
 }
 else
  printf("Stack is Empty \n");
}
void pop2 ()
{
 if(top2 < SIZE)
 {
  int popped_element = array[top2];
  top2++;
  printf("%d is being popped from Stack 2\n", popped_element);
 }
 else
  printf("Stack is Empty!\n");
}
void display_stack1 ()
{
 int i;
 for(i = top1; i >= 0; –i)
  printf("%d ", array[i]);
 printf("\n");
}
void display_stack2()
{
```

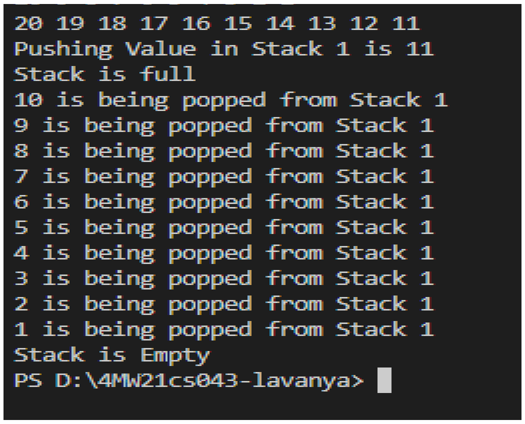


```
 int i;
 for(i = top2;i < SIZE; ++i)
  printf("%d ",array[i]);
 printf("\n");
}

int main() {
 int i;
 int num_of_ele;
 printf ("We can push a total of 20 values\n");
  for (i = 1; i <= 10; ++i)
  {
    push1(i);
    printf ("Value Pushed in Stack 1 is %d\n", i);
  }
  for (i = 11; i <= 20; ++i)
  {
    push2(i);
    printf ("Value Pushed in Stack 2 is %d\n", i);
  }
  display_stack1 ();
  display_stack2 ();
  printf ("Pushing Value in Stack 1 is %d\n", 11);
  push1 (11);
  num_of_ele = top1 + 1;
  while (num_of_ele)
  {
    pop1();
    --num_of_ele;
  }
  pop1();
  return 0;
}
```

Output



```
PS D:\4MW21cs043-lavanya> .\a.exe
We can push a total of 20 values
Value Pushed in Stack 1 is 1
Value Pushed in Stack 1 is 2
Value Pushed in Stack 1 is 3
Value Pushed in Stack 1 is 4
Value Pushed in Stack 1 is 5
Value Pushed in Stack 1 is 6
Value Pushed in Stack 1 is 7
Value Pushed in Stack 1 is 8
Value Pushed in Stack 1 is 9
Value Pushed in Stack 1 is 10
Value Pushed in Stack 2 is 11
Value Pushed in Stack 2 is 12
Value Pushed in Stack 2 is 13
Value Pushed in Stack 2 is 14
Value Pushed in Stack 2 is 15
Value Pushed in Stack 2 is 16
Value Pushed in Stack 2 is 17
Value Pushed in Stack 2 is 18
Value Pushed in Stack 2 is 19
Value Pushed in Stack 2 is 20
10 9 8 7 6 5 4 3 2 1
20 19 18 17 16 15 14 13 12 11
```

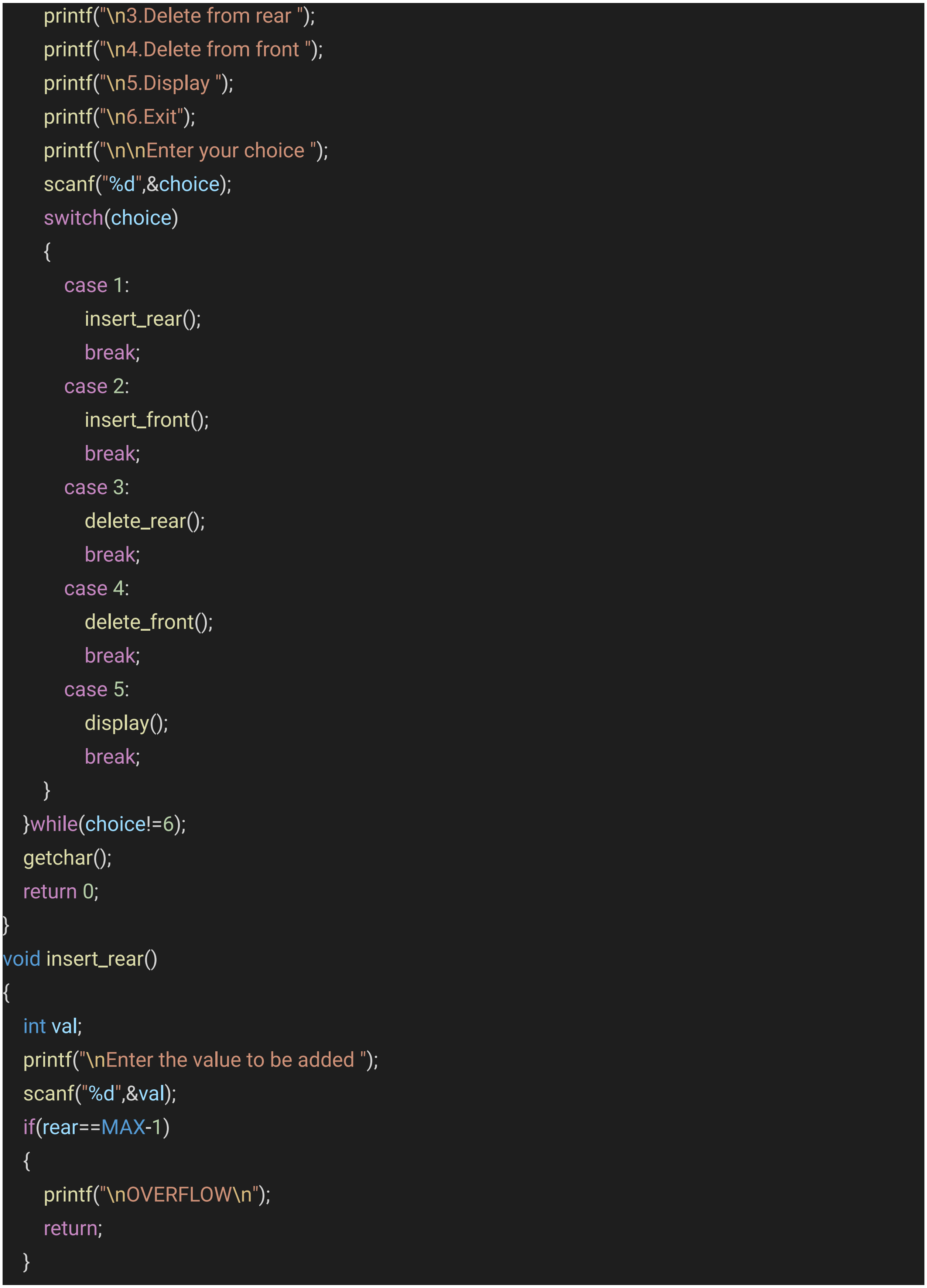


```
20 19 18 17 16 15 14 13 12 11
Pushing Value in Stack 1 is 11
Stack is full
10 is being popped from Stack 1
9 is being popped from Stack 1
8 is being popped from Stack 1
7 is being popped from Stack 1
6 is being popped from Stack 1
5 is being popped from Stack 1
4 is being popped from Stack 1
3 is being popped from Stack 1
2 is being popped from Stack 1
1 is being popped from Stack 1
Stack is Empty
PS D:\4MW21cs043-lavanya> 
```

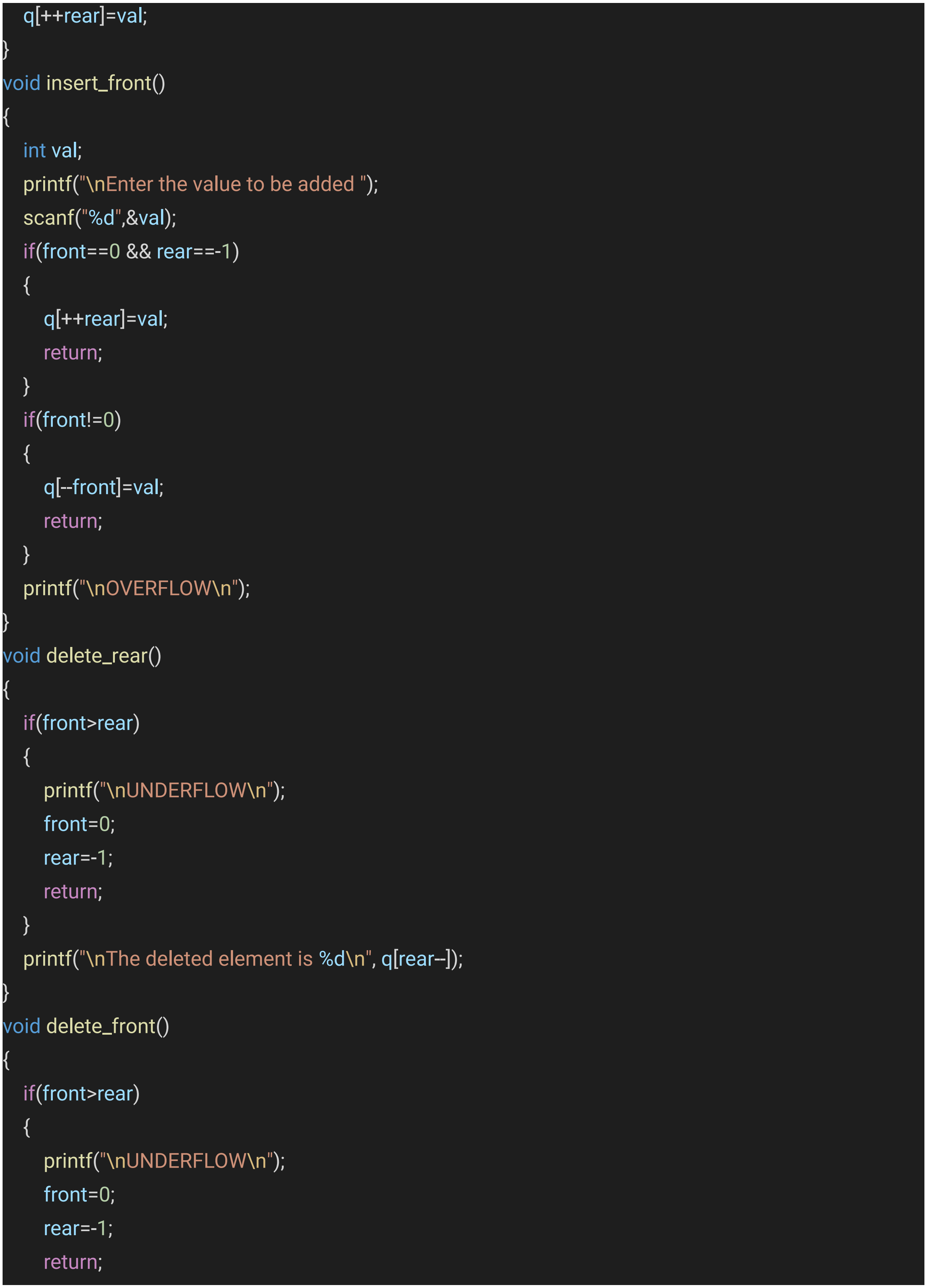
3) Develop a menu driven program to implement a double ended queue.

Program

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int q[MAX];
int front=0,rear=-1;
void insert_rear();
void insert_front();
void delete_rear();
void delete_front();
void display();
int main()
{
   int choice;
   do
   {
      printf("\n1.Insert at rear ");
      printf("\n2.Insert at front ");
```

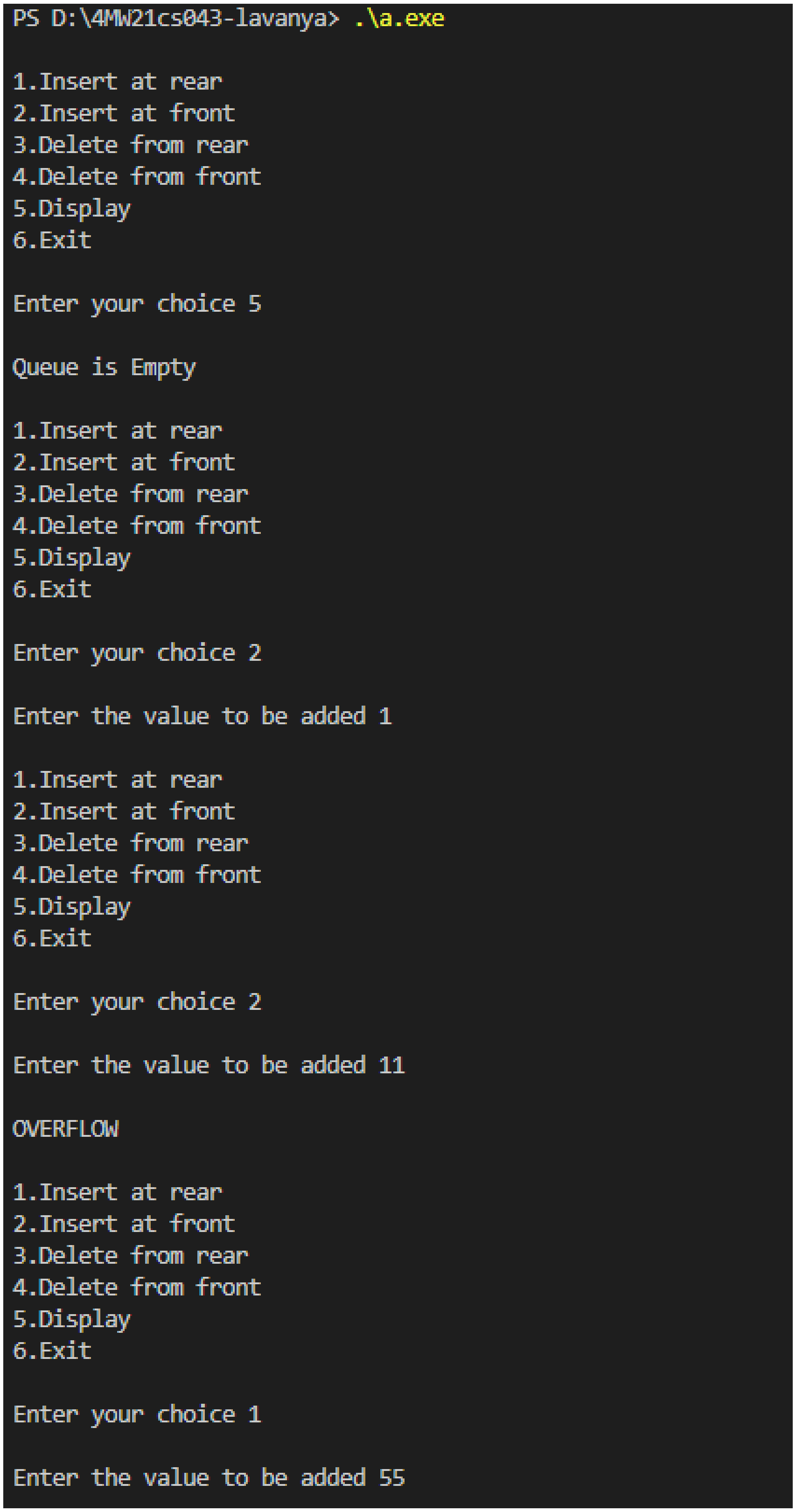


```
        printf("\n3.Delete from rear ");
        printf("\n4.Delete from front ");
        printf("\n5.Display ");
        printf("\n6.Exit");
        printf("\n\nEnter your choice ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                insert_rear();
                break;
            case 2:
                insert_front();
                break;
            case 3:
                delete_rear();
                break;
            case 4:
                delete_front();
                break;
            case 5:
                display();
                break;
        }
    }while(choice!=6);
    getchar();
    return 0;
}
void insert_rear()
{
    int val;
    printf("\nEnter the value to be added ");
    scanf("%d",&val);
    if(rear==MAX-1)
    {
        printf("\nOVERFLOW\n");
        return;
    }
```

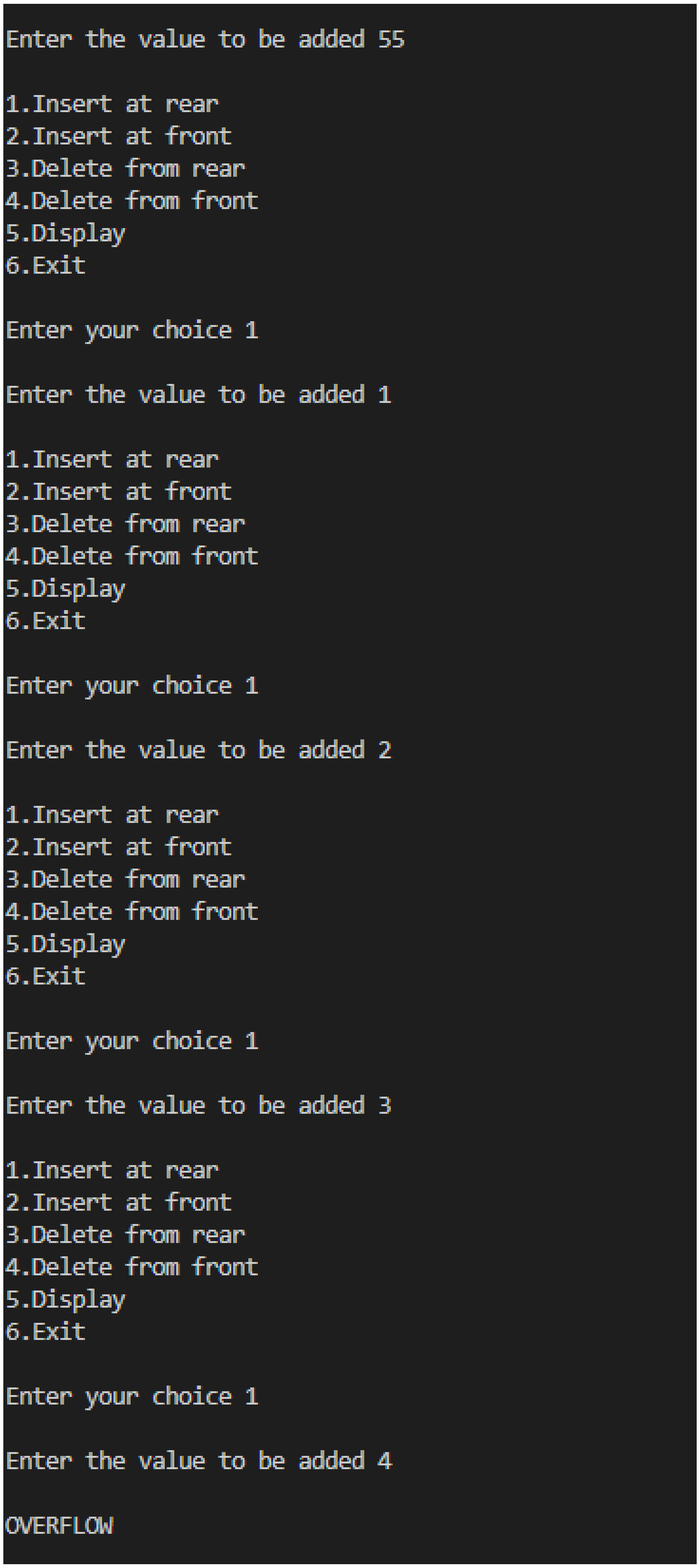


```
    q[++rear]=val;
}
void insert_front()
{
    int val;
    printf("\nEnter the value to be added ");
    scanf("%d",&val);
    if(front==0 && rear==-1)
    {
        q[++rear]=val;
        return;
    }
    if(front!=0)
    {
        q[--front]=val;
        return;
    }
    printf("\nOVERFLOW\n");
}
void delete_rear()
{
    if(front>rear)
    {
        printf("\nUNDERFLOW\n");
        front=0;
        rear=-1;
        return;
    }
    printf("\nThe deleted element is %d\n", q[rear--]);
}
void delete_front()
{
    if(front>rear)
    {
        printf("\nUNDERFLOW\n");
        front=0;
        rear=-1;
        return;
    }
    printf("\nThe deleted element is %d\n", q[front++]);
}
void display()
{
    if(front>rear)
    {
        printf("\nQueue is Empty\n");
        return;
    }
    printf("\nThe elements in the queue are: ");
    for (int i = front; i<=rear;i++)
        printf("%d ",q[i]);
}
```

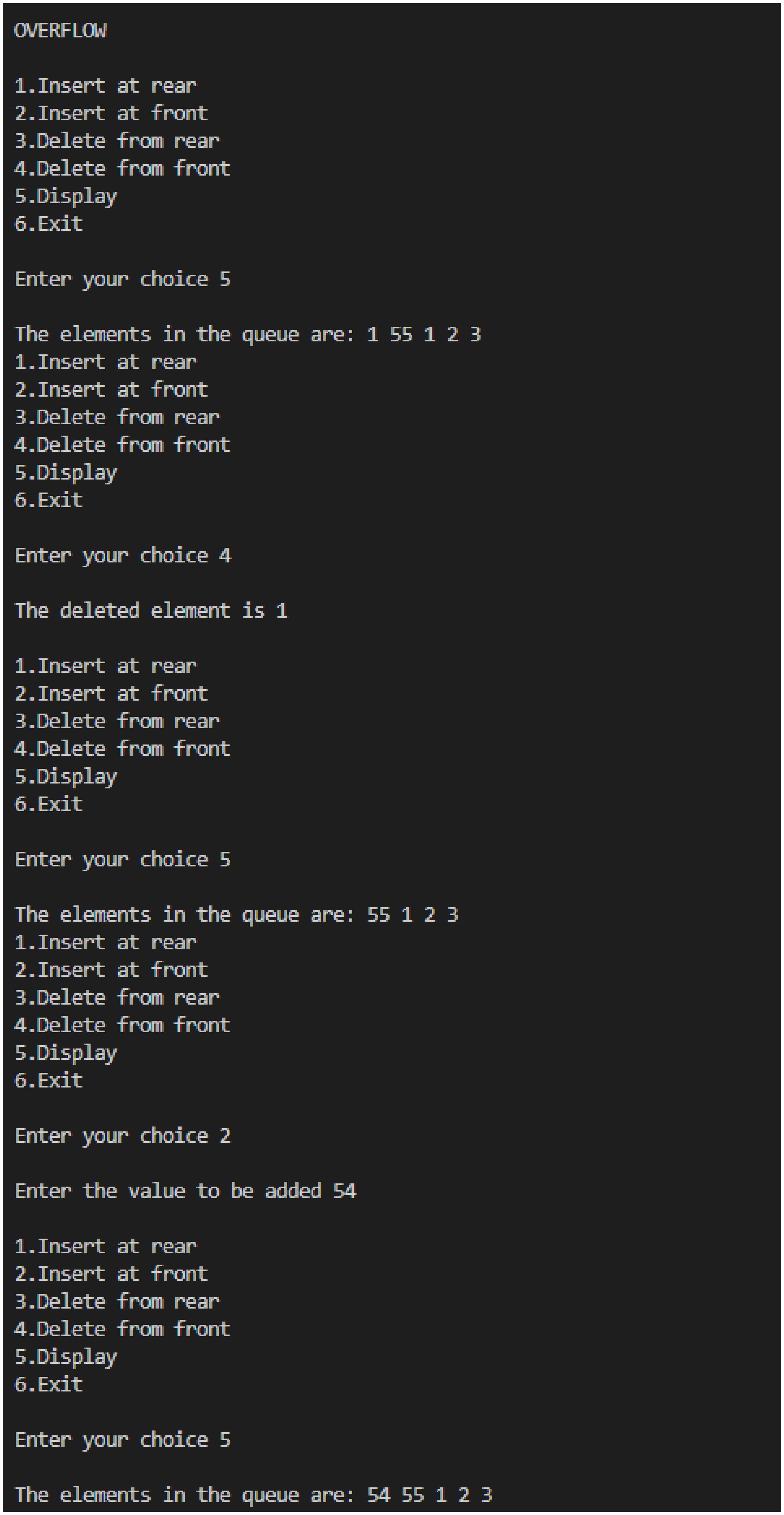
Output



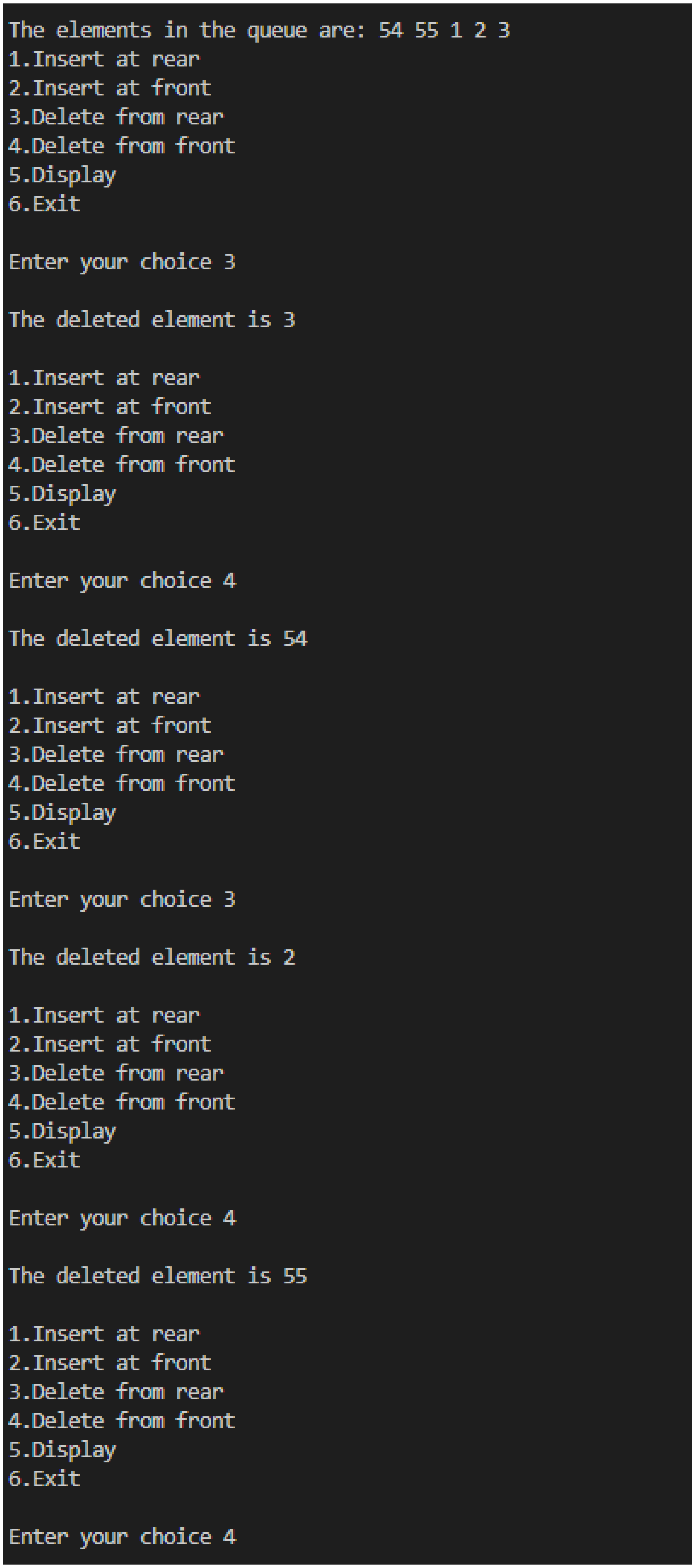
```
PS D:\4MW21cs043-lavanya> .\a.exe

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 5

Queue is Empty

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 2

Enter the value to be added 1

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 2

Enter the value to be added 11

OVERFLOW

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 1

Enter the value to be added 55
```



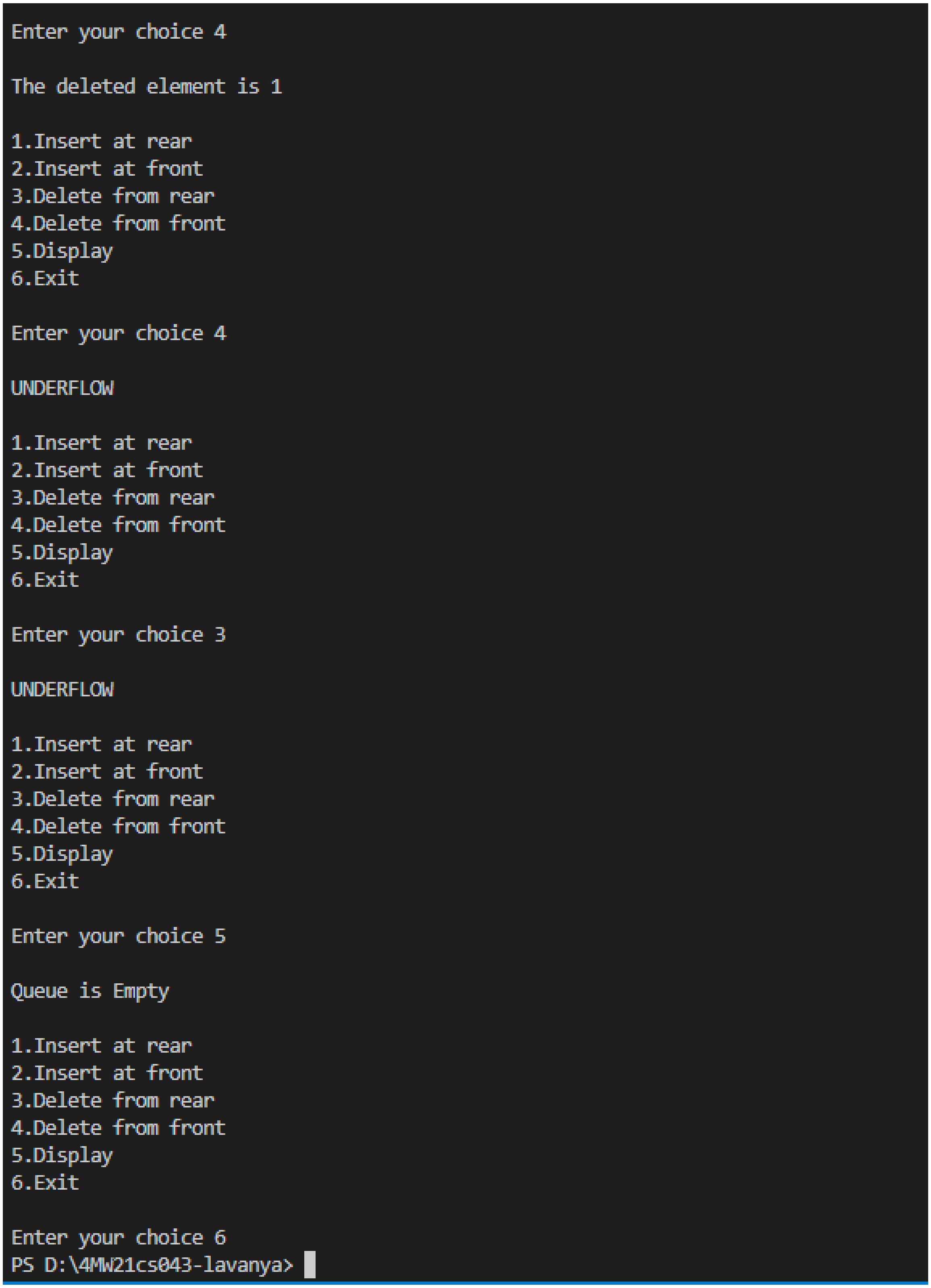
```
Enter the value to be added 55

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 1

Enter the value to be added 1

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 1

Enter the value to be added 2

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 1

Enter the value to be added 3

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 1

Enter the value to be added 4

OVERFLOW
```



```
OVERFLOW

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 5

The elements in the queue are: 1 55 1 2 3
1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 4

The deleted element is 1

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 5

The elements in the queue are: 55 1 2 3
1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 2

Enter the value to be added 54

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 5

The elements in the queue are: 54 55 1 2 3
```



```
The elements in the queue are: 54 55 1 2 3
1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 3

The deleted element is 3

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 4

The deleted element is 54

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 3

The deleted element is 2

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 4

The deleted element is 55

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 4
```



```
Enter your choice 4

The deleted element is 1

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 4

UNDERFLOW

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 3

UNDERFLOW

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 5

Queue is Empty

1.Insert at rear
2.Insert at front
3.Delete from rear
4.Delete from front
5.Display
6.Exit

Enter your choice 6
PS D:\4MW21cs043-lavanya>
```

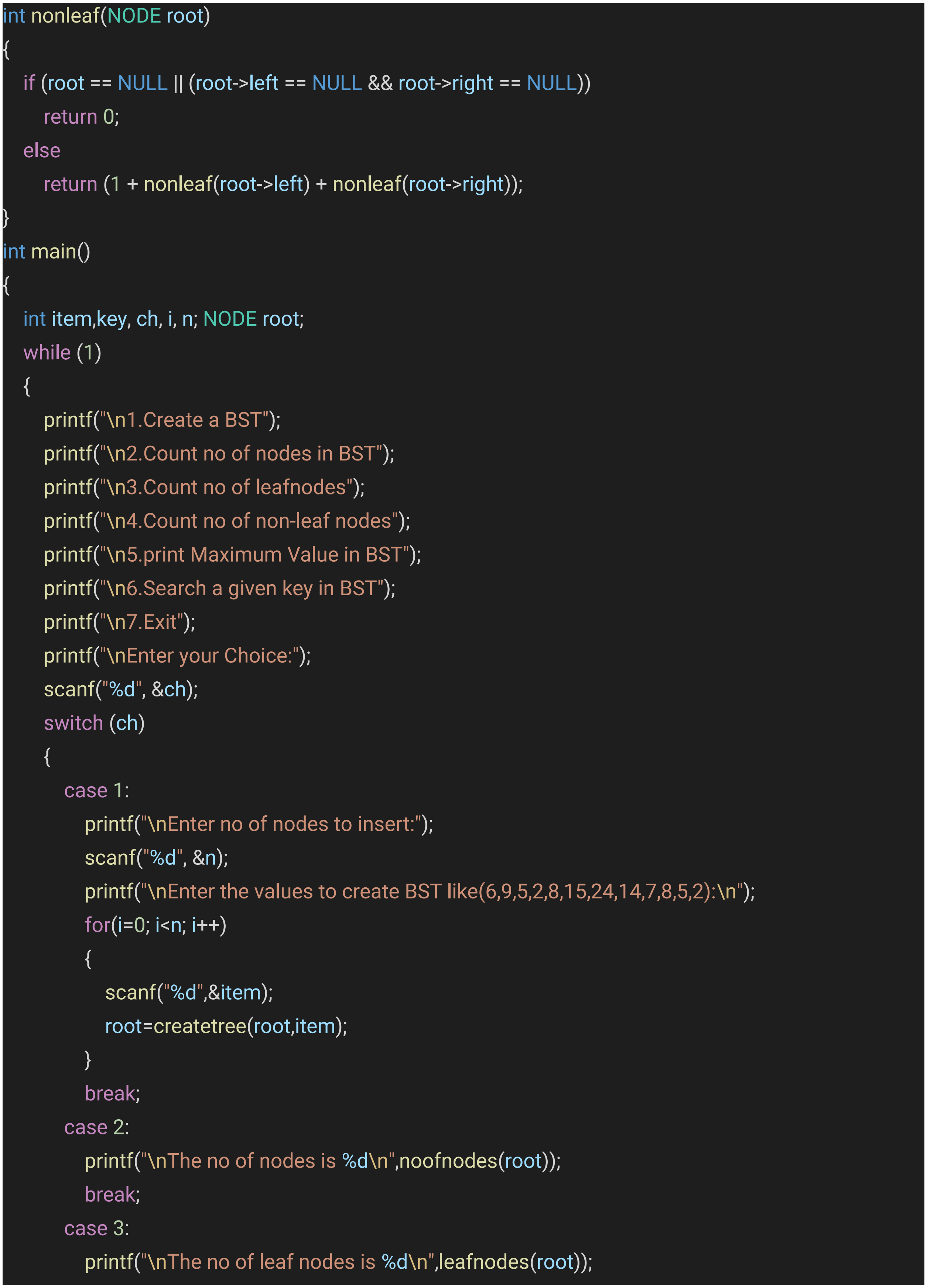
4)Develop a menu driven program to implement following operations on Binary Search Tree.

1. To count the number of nodes in the BST.
2. To count the number of leaf nodes in the BST.
3. To count the number of non-leaf nodes in the BST.
4. To print the maximum value in the given BST.
5. To search for a given key value in the BST.

Program

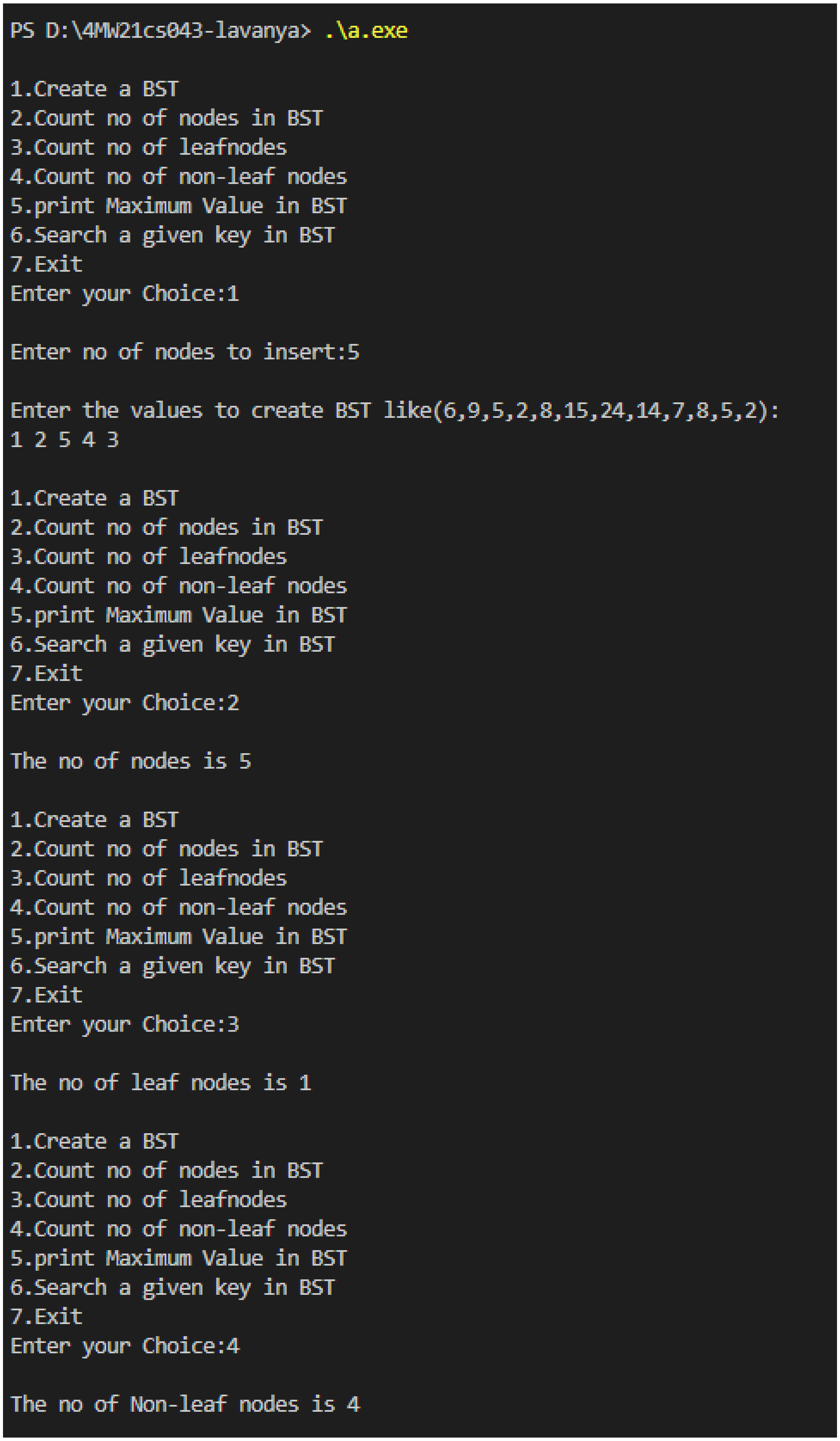
```
#include <stdio.h>
#include <stdlib.h>
struct BST
{
   int data;
   struct BST *left;
   struct BST *right;
};
typedef struct BST* NODE;
NODE createtree(NODE root, int item)
{
   if (root == NULL)
   {
      NODE root;
      root=(NODE)malloc(sizeof(struct BST));
      root->data = item;
      root->left = root->right = NULL;
      return root;
   }
   if (item < (root->data))
      root->left = createtree(root->left, item);
   else if (item > root->data)
      root -> right = createtree(root->right, item);
   return root;
}
int search(NODE root, int key)
{
   if(root == NULL)
      return (printf("%d is not present\n",key));
   else
```

```
    {
        if(key==root->data)

            return (printf("%d is present\n",key));

        else if(key>root->data)
            search(root->right,key);
        else
        search(root->left,key);
    }
}
int noofnodes(NODE root)
{
    int c=0;
    if(root==NULL)
        return c;
    c++;
    c = c + noofnodes(root->right);
    c = c + noofnodes(root->left);
    return c;
}
int leafnodes(NODE root)
{
    int c=0;
    if(root==NULL)
        return c;
    if(root->right==NULL && root->left==NULL)
        c++;
    c=c+leafnodes(root->right);
    c=c+leafnodes(root->left);
    return c;
}
int MAX(NODE root)
{
    if(root->right==NULL)
        return root->data;
    MAX(root->right);
}
```

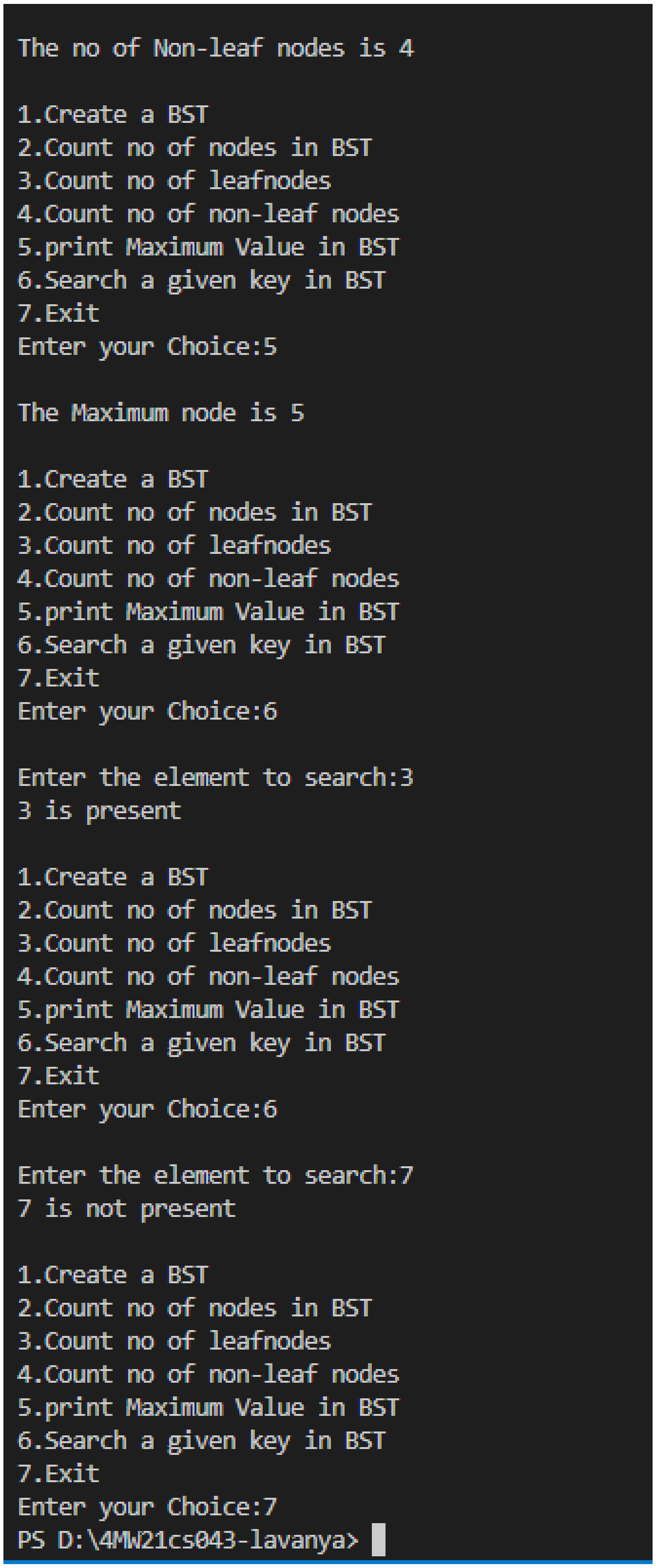


```
int nonleaf(NODE root)
{
    if (root == NULL || (root->left == NULL && root->right == NULL))
        return 0;
    else
        return (1 + nonleaf(root->left) + nonleaf(root->right));
}
int main()
{
    int item,key, ch, i, n; NODE root;
    while (1)
    {
        printf("\n1.Create a BST");
        printf("\n2.Count no of nodes in BST");
        printf("\n3.Count no of leafnodes");
        printf("\n4.Count no of non-leaf nodes");
        printf("\n5.print Maximum Value in BST");
        printf("\n6.Search a given key in BST");
        printf("\n7.Exit");
        printf("\nEnter your Choice:");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                printf("\nEnter no of nodes to insert:");
                scanf("%d", &n);
                printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2):\n");
                for(i=0; i<n; i++)
                {
                    scanf("%d",&item);
                    root=createtree(root,item);
                }
                break;
            case 2:
                printf("\nThe no of nodes is %d\n",noofnodes(root));
                break;
            case 3:
                printf("\nThe no of leaf nodes is %d\n",leafnodes(root));
                break;
            case 4:
                printf("\nThe no of Non-leaf nodes is %d\n",nonleaf(root));
                break;
            case 5:
                printf("\nThe Maximum node is %d\n",MAX(root));
                break;
            case 6:
                printf("\nEnter the element to search:"); scanf("%d", &key);
                search(root,key);
                break;
            case 7:
                exit(0);
            default:
                printf("\nInvalid Choice");
                break;
            return 0;
        }
    }
}
```

<u>**Output**</u>



```
PS D:\4MW21cs043-lavanya> .\a.exe

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:1

Enter no of nodes to insert:5

Enter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2):
1 2 5 4 3

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:2

The no of nodes is 5

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:3

The no of leaf nodes is 1

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:4

The no of Non-leaf nodes is 4
```

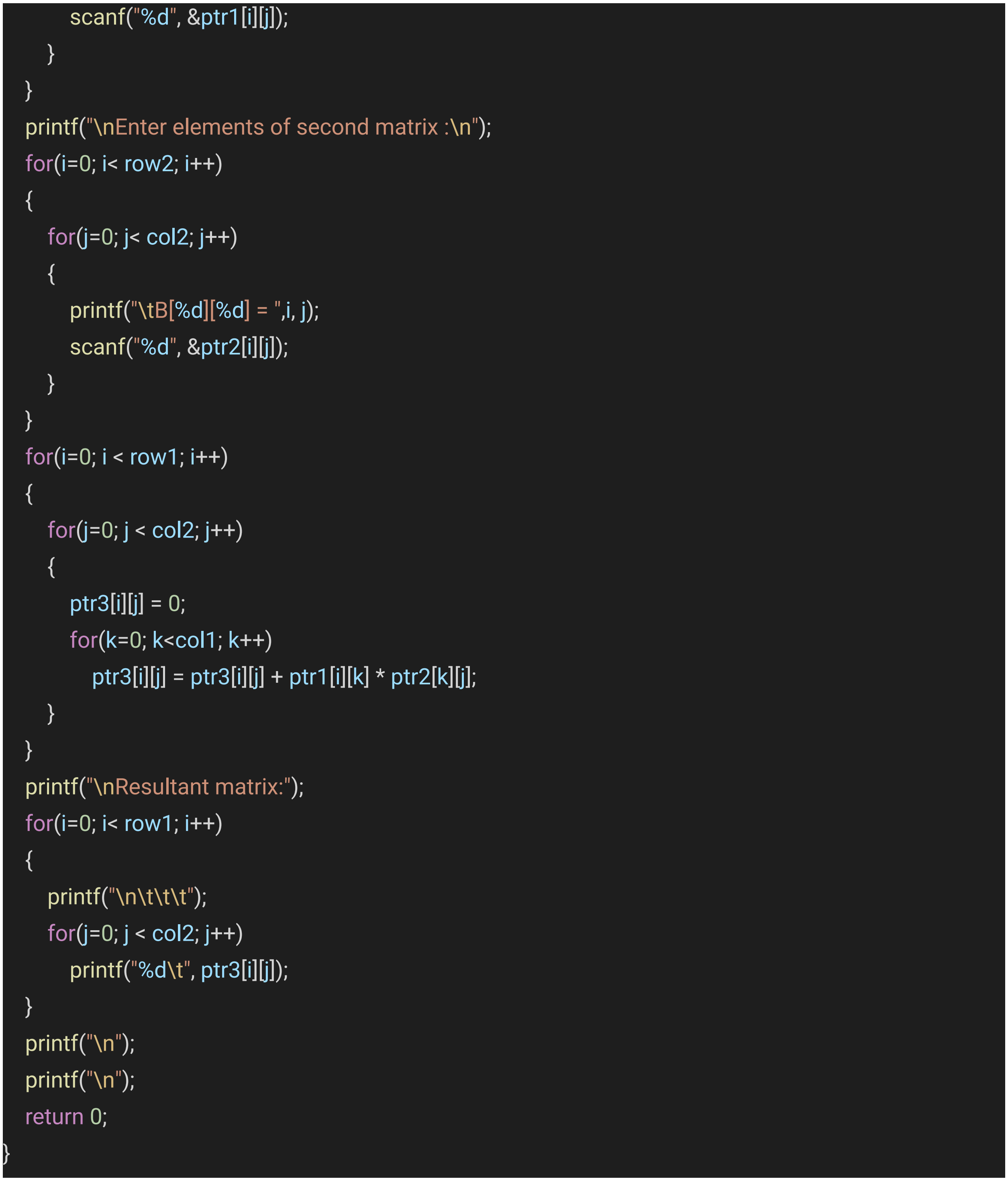


```
The no of Non-leaf nodes is 4

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:5

The Maximum node is 5

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:6

Enter the element to search:3
3 is present

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:6

Enter the element to search:7
7 is not present

1.Create a BST
2.Count no of nodes in BST
3.Count no of leafnodes
4.Count no of non-leaf nodes
5.print Maximum Value in BST
6.Search a given key in BST
7.Exit
Enter your Choice:7
PS D:\4MW21cs043-lavanya>
```

5) Develop a program to multiply two matrices using dynamically allocated arrays

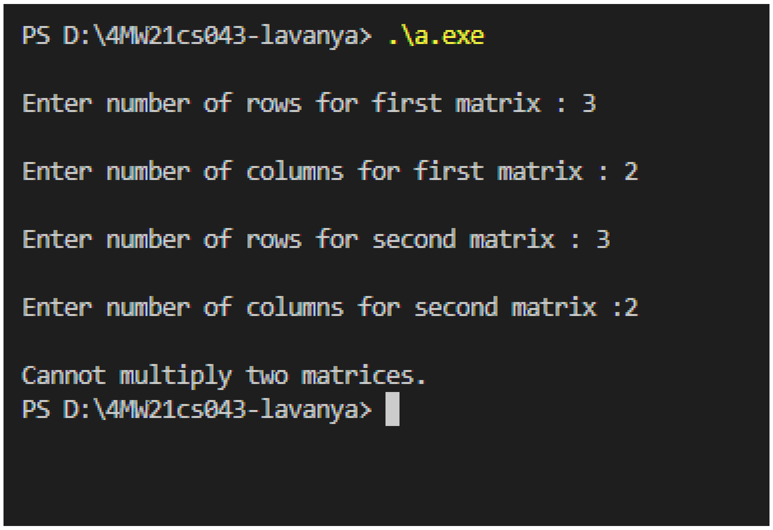
Program

```
#include <stdio.h>
#include<stdlib.h>
int main()
{
   int **ptr1, **ptr2, **ptr3;
   int row1, col1, row2, col2;
   int i, j, k;
   printf("\nEnter number of rows for first matrix : ");
   scanf("%d", &row1);
   printf("\nEnter number of columns for first matrix : ");
   scanf("%d", &col1);
   printf("\nEnter number of rows for second matrix : ");
   scanf("%d", &row2);
   printf("\nEnter number of columns for second matrix :");
   scanf("%d", &col2);
   if(col1 != row2)
   {
      printf("\nCannot multiply two matrices.");
      return(0);
   }
   ptr1 = (int **) malloc(sizeof(int *) * row1);
   ptr2 = (int **) malloc(sizeof(int *) * row2);
   ptr3 = (int **) malloc(sizeof(int *) * row1);
   for(i=0; i<row1; i++)
      ptr1[i] = (int *)malloc(sizeof(int) * col1);
   for(i=0; i<row2; i++)
      ptr2[i] = (int *)malloc(sizeof(int) * col2);
   for(i=0; i<row1; i++)
      ptr3[i] = (int *)malloc(sizeof(int) * col2);
   printf("\nEnter elements of first matrix :\n");
   for(i=0; i< row1; i++)
   {
      for(j=0; j< col1; j++)
      {
         printf("\tA[%d][%d] = ",i, j);
```

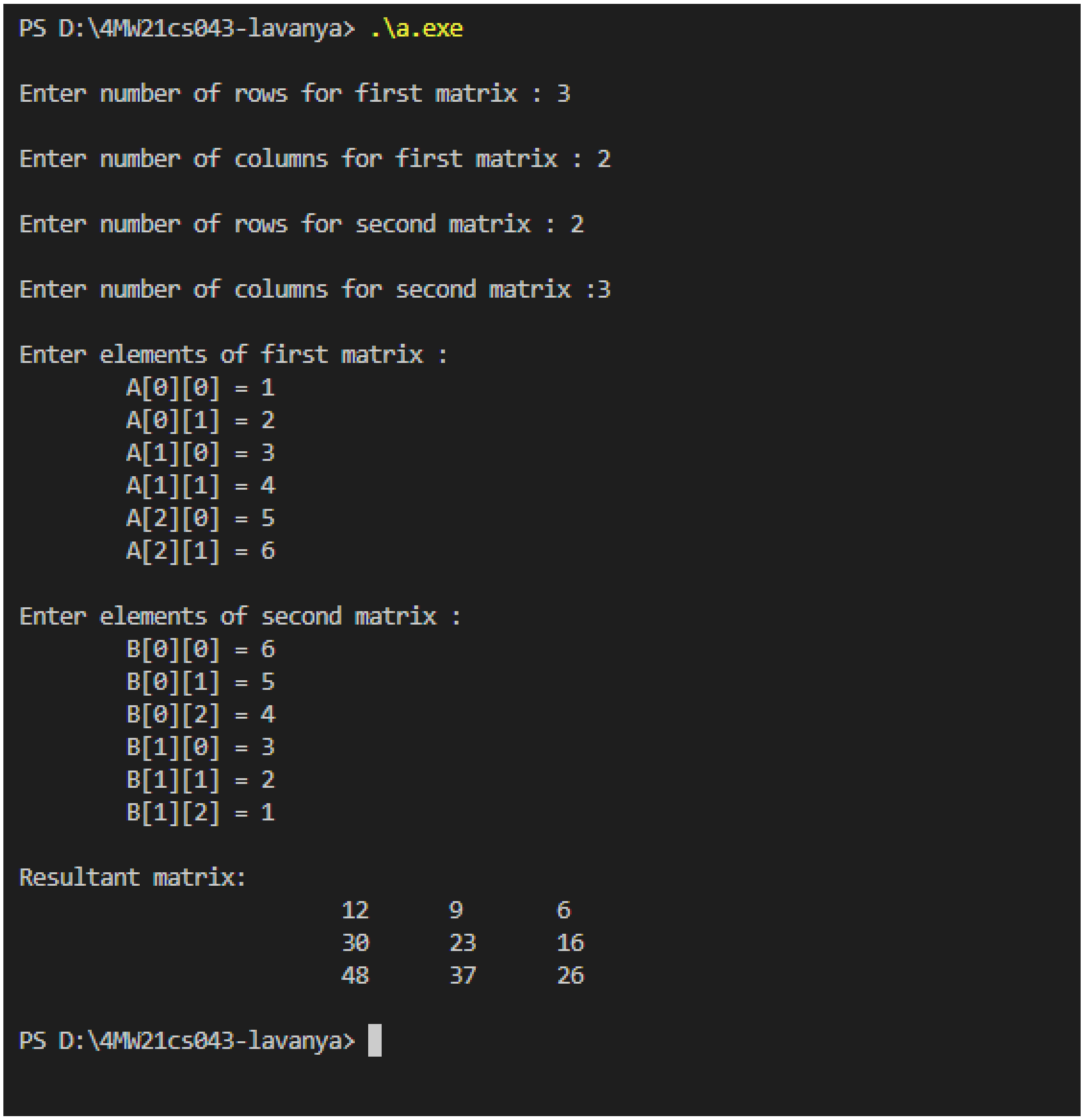


```
            scanf("%d", &ptr1[i][j]);
        }
    }
    printf("\nEnter elements of second matrix :\n");
    for(i=0; i< row2; i++)
    {
        for(j=0; j< col2; j++)
        {
            printf("\tB[%d][%d] = ",i, j);
            scanf("%d", &ptr2[i][j]);
        }
    }
    for(i=0; i < row1; i++)
    {
        for(j=0; j < col2; j++)
        {
            ptr3[i][j] = 0;
            for(k=0; k<col1; k++)
                ptr3[i][j] = ptr3[i][j] + ptr1[i][k] * ptr2[k][j];
        }
    }
    printf("\nResultant matrix:");
    for(i=0; i< row1; i++)
    {
        printf("\n\t\t\t");
        for(j=0; j < col2; j++)
            printf("%d\t", ptr3[i][j]);
    }
    printf("\n");
    printf("\n");
    return 0;
}
```

Output



```
PS D:\4MW21cs043-lavanya> .\a.exe

Enter number of rows for first matrix : 3

Enter number of columns for first matrix : 2

Enter number of rows for second matrix : 3

Enter number of columns for second matrix :2

Cannot multiply two matrices.
PS D:\4MW21cs043-lavanya> 
```



```
PS D:\4MW21cs043-lavanya> .\a.exe

Enter number of rows for first matrix : 3

Enter number of columns for first matrix : 2

Enter number of rows for second matrix : 2

Enter number of columns for second matrix :3

Enter elements of first matrix :
        A[0][0] = 1
        A[0][1] = 2
        A[1][0] = 3
        A[1][1] = 4
        A[2][0] = 5
        A[2][1] = 6

Enter elements of second matrix :
        B[0][0] = 6
        B[0][1] = 5
        B[0][2] = 4
        B[1][0] = 3
        B[1][1] = 2
        B[1][2] = 1

Resultant matrix:
                        12      9       6
                        30      23      16
                        48      37      26

PS D:\4MW21cs043-lavanya> 
```