GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA-141006

EXPERIMENT 1 Program to insert a new element.

1. At end as well as
2. At a given position of an array

CODE:

#include<iostream> using namespace std; int main(){

int size=5;

int arr[size] = {45,25,65,95,85}; cout << "Elements in Array : \n"; for(int i=0;i<size;i++){ cout << arr[i] << "\t";

}

cout << endl << endl; int newElement; cout << "Enter New Element : "; cin >> newElement; cout << endl;

cout << "Choices for Insertion : \n";

cout << "1. You want insert element at Particular Postion or \n"; cout << "2. Insert as Last Element in List.....\n"<<endl; int choice;

cout << "Please Select Choice for Insertion : "; cin >> choice; cout << endl; if(choice==1)

{

int elemnum;

cout << "Enter the number (element number) Which u want to insert

:";

cin >> elemnum; int pos;

pos=elemnum-1;

cout << endl; int j;

for(j=size-1;j>=pos;j--)

{ arr[j+1] = arr[j];

}

arr[pos] = newElement; size = size+1;

for(int i=0;i<size;i++)

{ cout << arr[i] << "\t";

}

}

else if(choice==2)

{

int lastpos=size; arr[lastpos] = newElement; size=lastpos+1;

for(int i=0;i<size;i++)

{ cout << arr[i] << "\t";

}

}

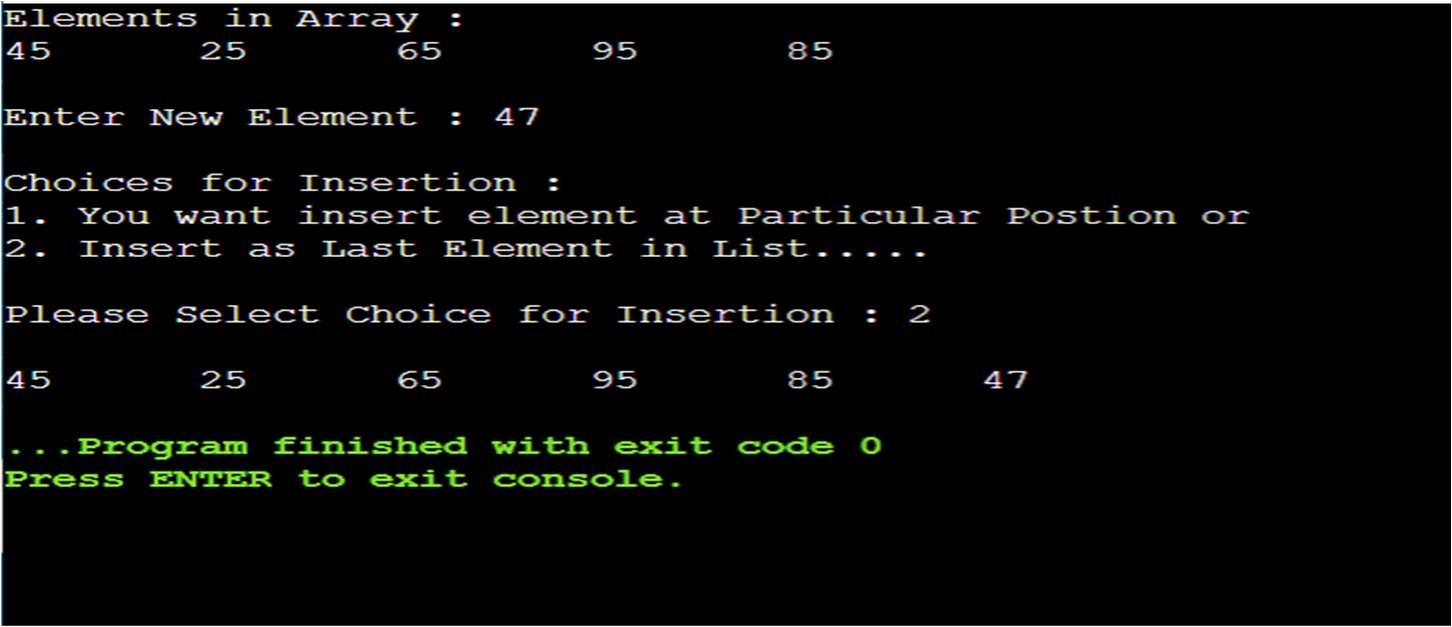
else

cout << "You Have not entered any Choice"; return 0

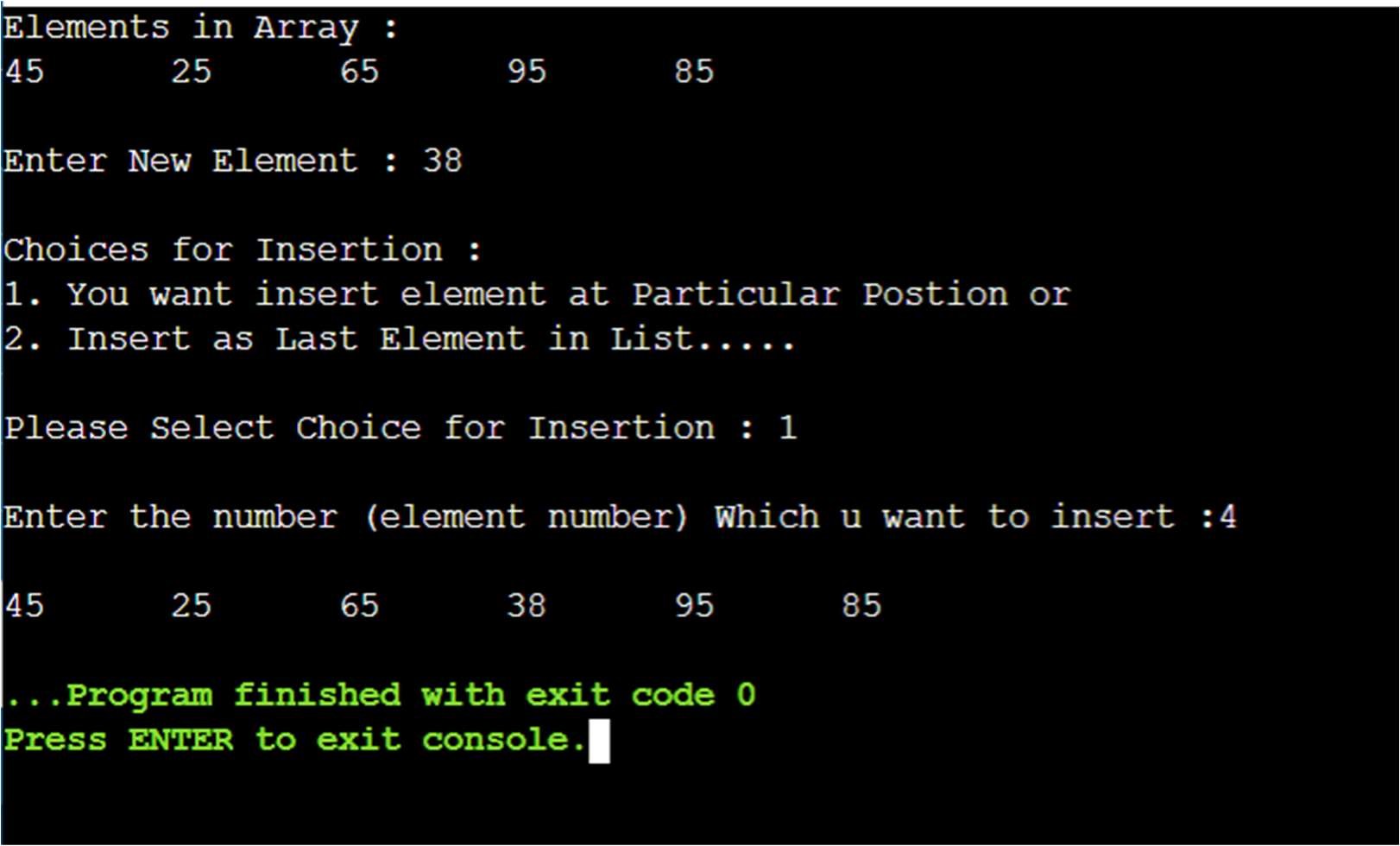
}

OUTPUT:

1. ELEMENT INSERTED AT LAST



1. ELEMENT INSERTED AT PARTICULAR POSITION



EXPERIMENT 2 Program to delete an element

CODE:

#include<iostream> using namespace std; int main()

{

int size=5;

int arr[size] = {45,25,65,95,85}; cout << "Elements in Array : \n"; for(int i=0; i<size; i++)

{

cout << arr[i] << "\t";

}

cout << endl << endl; cout << "Choices for Deletion : \n";

cout << "1. Enter the Element which u want to delete or \n"; cout << "2. Enter the position of element.....\n"<<endl; int choice;

cout << "Please Select Choice for Deletion : "; cin >> choice; cout << endl; if(choice==1)

{

int delElement;

cout << "Enter the element to be deleted : "; cin >> delElement; for(int i=0;i<size;i++)

{

if(arr[i]==delElement

{

for(int j=i;j<size;j++)

{

arr[j]=arr[j+1];

}

break;

}

}

for(int i=0;i<(size-1);i++)

{

cout<<arr[i]<<"\t";

}

}

else if(choice==2) { int position;

cout << "Enter position of element u want to delete : "; cin >> position; int elemnum; elemnum=position-1; cout <<endl; int item = arr[elemnum]; int j = size; for(j=elemnum;j<size;j++){ arr[j] = arr[j+1]

}

size = size-1; for(int i=0;i<size;i++)

{ cout << arr[i] << "\t";

}

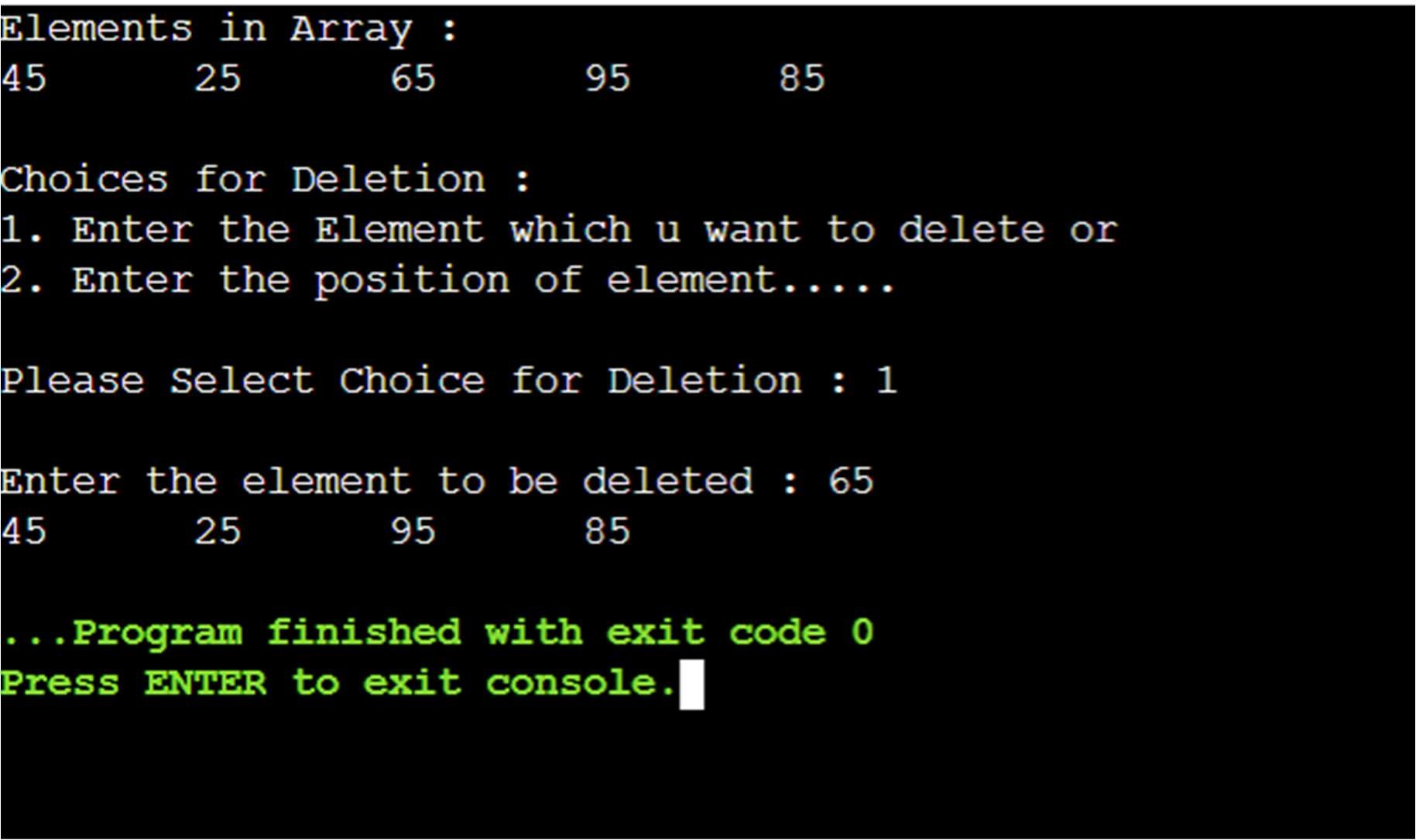
}

else

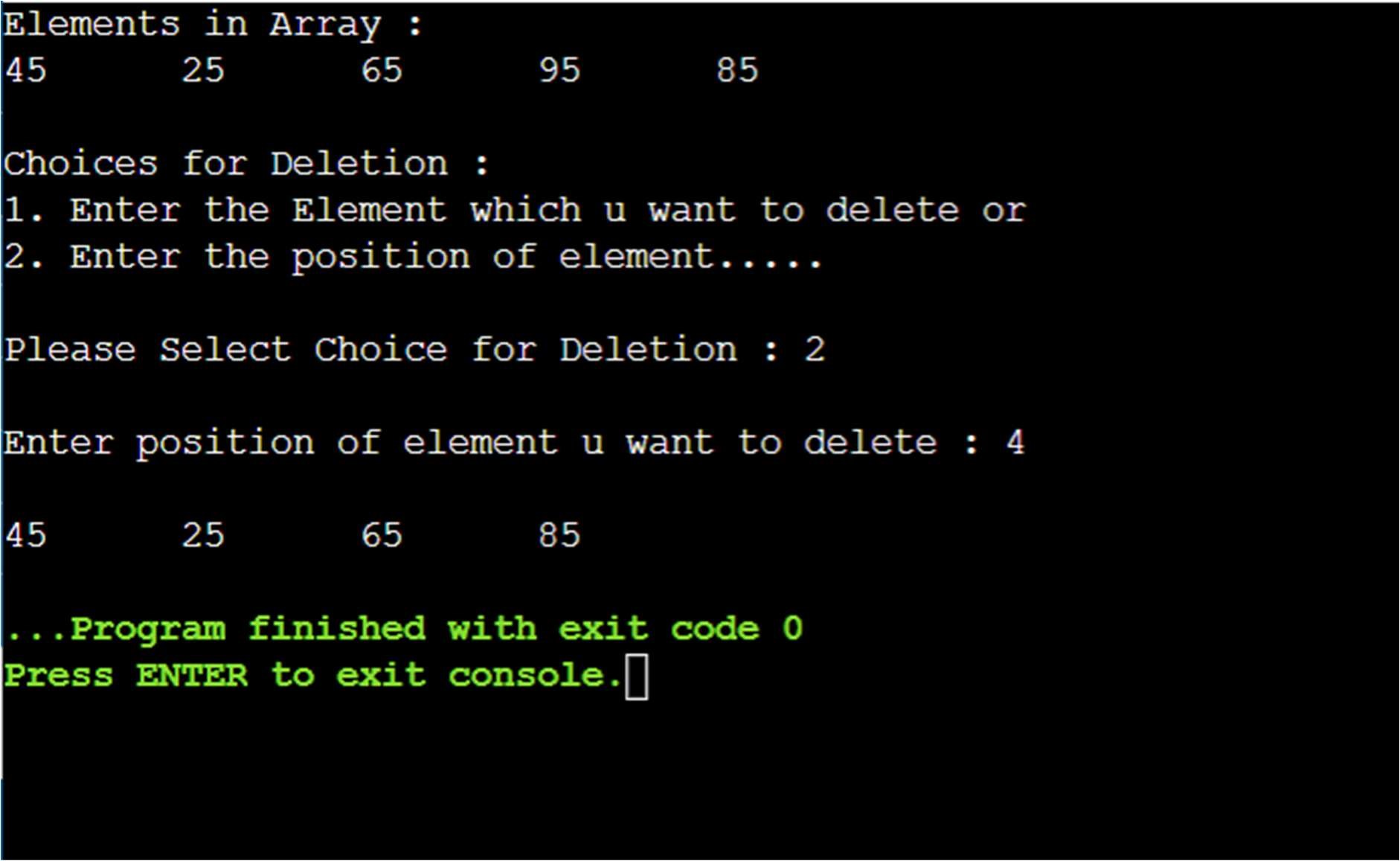
cout << "You Have not entered any Choice"; return 0;

}

OUTPUT: (1) ELEMENTS WHOSE VALUE IS GIVEN



(2) ELEMENT WHOSE POSITION IS GIVEN



Experiment 3: Program to find the location of a give element using Linear Search.

CODE:

#include<iostream> using namespace std;

int linearsearch(int arr[],int size, int searchElement)

{ for(int i=0;i<=size-1;i++) if(arr[i]==searchElement)

{

int searched\_at=(i+1); return searched\_at;

}

return -1;

} int main()

{

int size = 10;

int arr[size] = {2,5,6,6,12,15,25,36,49,55}; cout << "Elements in array : \n"; for(int i=0;i<10;i++)

{

cout << arr[i] << "\t";

}

int searchElement;

cout << "\nEnter the Element you want to search in the array (using linear search) : "; cin >> searchElement;

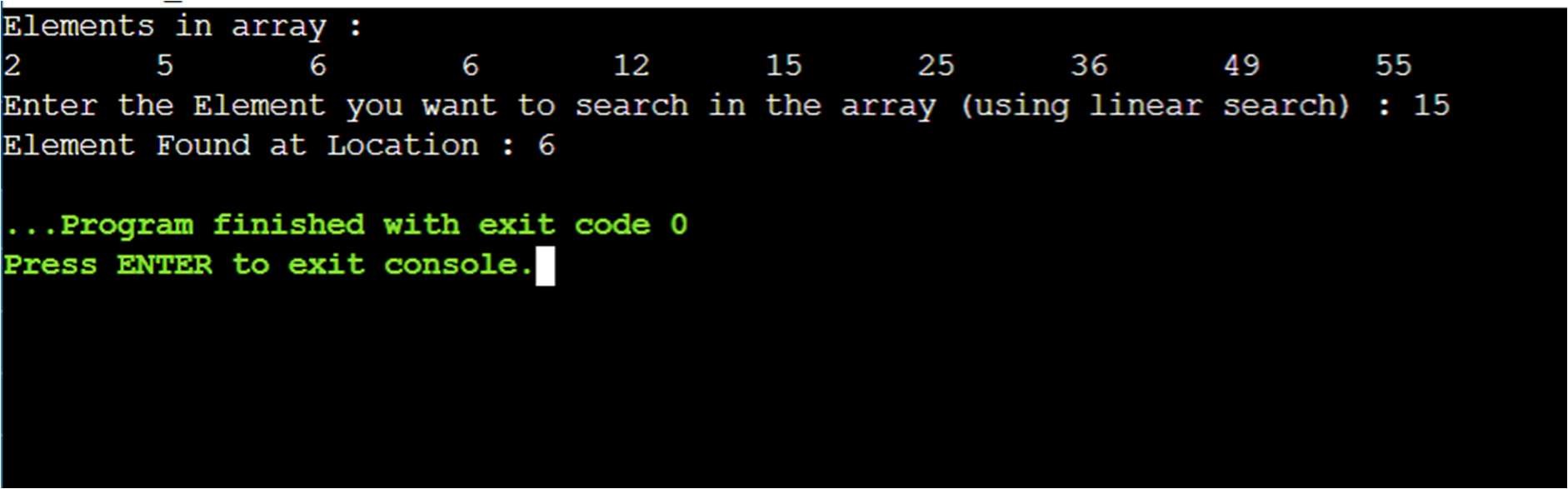
int location = linearsearch(arr,size,searchElement) if(location == -1)

cout << "Element Not Found"; else

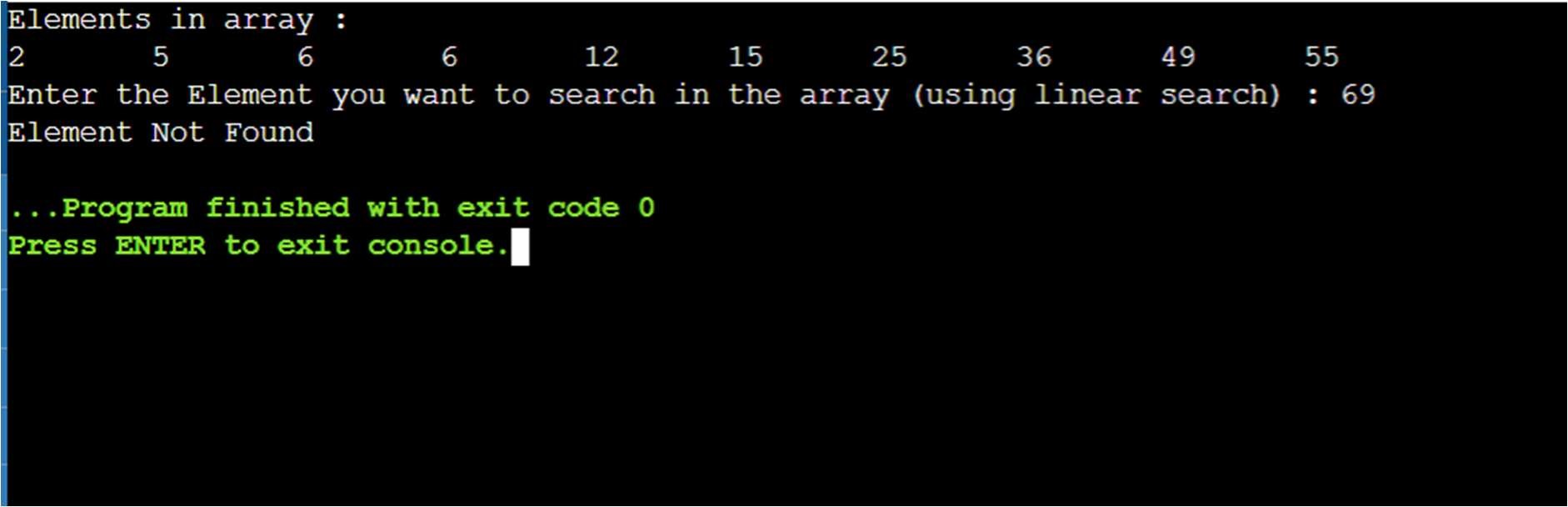
cout << "Element Found at Location in the array (element number in array)): " << location; return 0;

}

OUTPUT: (1) ELEMENT FOUND AT PARTICULAR POSITON



(2) ELEMENT NOT FOUND IN ARRAY



Experiment 4: Program to find the location of a given element using Binary Search.

CODE:

#include<iostream> using namespace std;

int binarysearch(int data[],int lb, int ub, int item)

{

int beg=lb; int end=ub; int loc;

int mid = (beg+end)/2;

cout<<"\n mid = "<<mid; cout<<"\n value at mid is = "<<data[mid]; while((beg<=end)&& (data[mid]!=item))

{

if(item<data[mid])

{

end=mid-1;

} else { beg=mid+1;

}

mid=(beg+end)/2;

}

if (data[mid]==item)

{

loc=mid;

}

else

{

loc=-1;

}

return loc;

}

int main()

{

int size = 11;

int data[size] = {2,5,6,7,12,15,25,36,49,55,66}; cout << "\n Elements in array : \n"; for(int i=0;i<10;i++)

{

cout << data[i] << "\t";

}

int item, location;

cout << "\nEnter the Element that you want to search using binary search: "; cin >> item;

location = binarysearch(data,0,size-1,item); if(location==-1)

{

cout<<"\n Element is not found";

}

else

{

int elemnum=location+1;

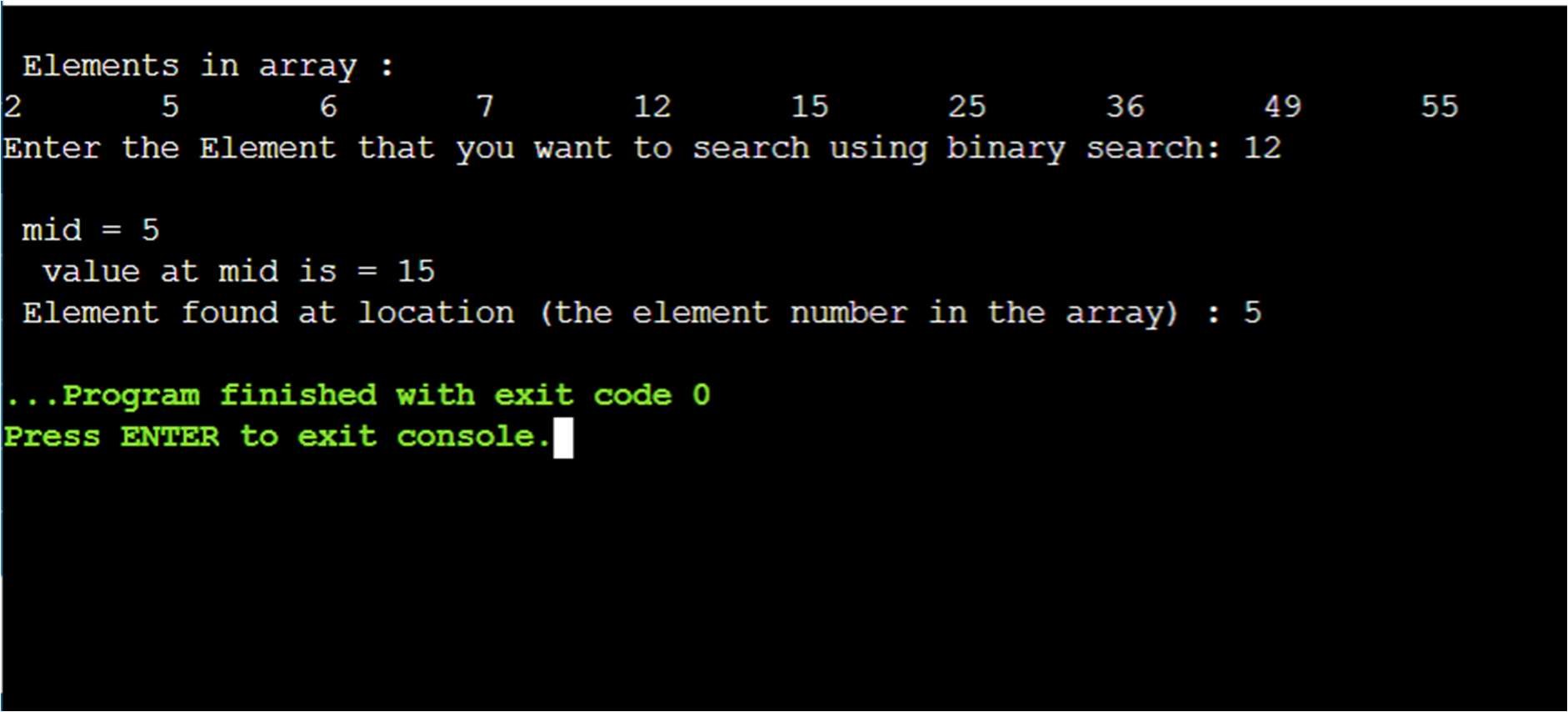
cout << "\n Element found at location (the element number in the array) : " << elemnum;

}

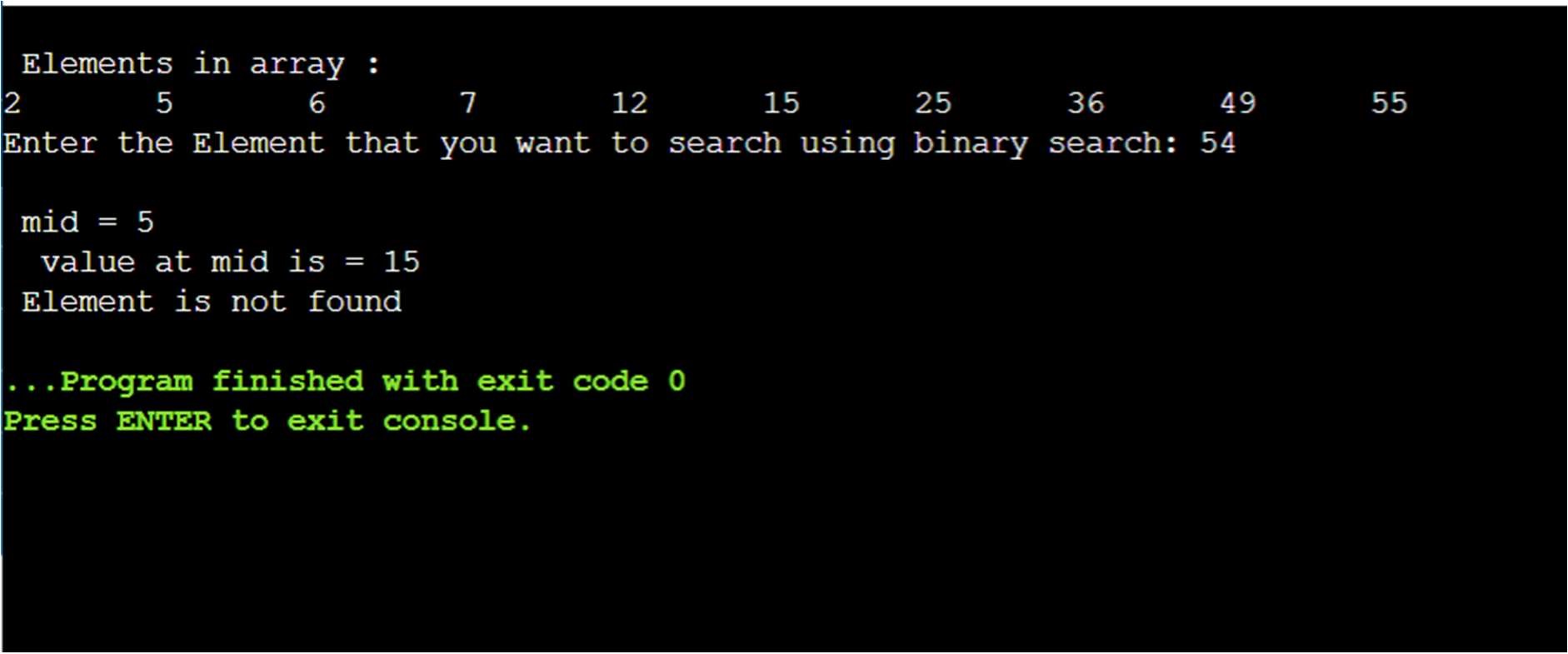
return 0;

}

1. ELEMENT AT PARTICULAR POSITION



1. ELEMEMT NOT FOUND



Experiment 5: Program to implement push and pop operations on a stack using linear array.

CODE:

#include<iostream> #define size 5 using namespace std;

int top=-1,stack[size]; void push(int newElement)

{ if(top==size-1)

{

cout << "\n.....................\n";

cout << "Stack is Full.....cannot insert this value......\n"; cout << "\n..............................\n";

}

else{ top=top+1; stack[top]= newElement;

} }

void pop()

{

if(top==-1)

{

cout << "\n..............................\n"; cout << "\nStack is empty!!";

cout << "\n..............................\n";

}

else{

cout << "\n..............................\n";

cout << "\n..........deleting the elemnt....................\n"; cout<<"\n..\n"; cout<<"\n..\n"; cout<<"\n..\n"; cout<<"\n..\n"; cout<<"\n..\n";

cout <<"\nDeleted element is " << stack[top]; top=top-1; cout<<"\n..\n"; cout<<"\n..\n"; cout<<"\n..\n"; cout<<"\n..\n"; cout<<"\n..\n";

cout << "\n..............................\n";

}

}

void display()

{ int i;

if(top==-1)

{

cout <<"\nStack is empty!!";

}

else{

cout << "\nStack is...(Maximun capacity as 5 elements)\n"; cout << "\n..............................\n"; for(i=top;i>=0;--i) cout << stack[i] <<"\n";

}

cout << "\n..............................\n"; cout << endl;

}

int main(){ int choice; int newElement; while(1){

cout << "\nSelect Your Choice : ........"; cout << "1.Push 2. Pop 3.Display 4.Exit" <<endl; cout << "\nEnter Your Choice : "; cin >> choice; cout << endl; switch(choice)

{

case 1:

{

cout << "\n..............................\n"; cout << "Enter Element : "; cin >> newElement;

cout << "\n..............................\n"; push(newElement); break;

}

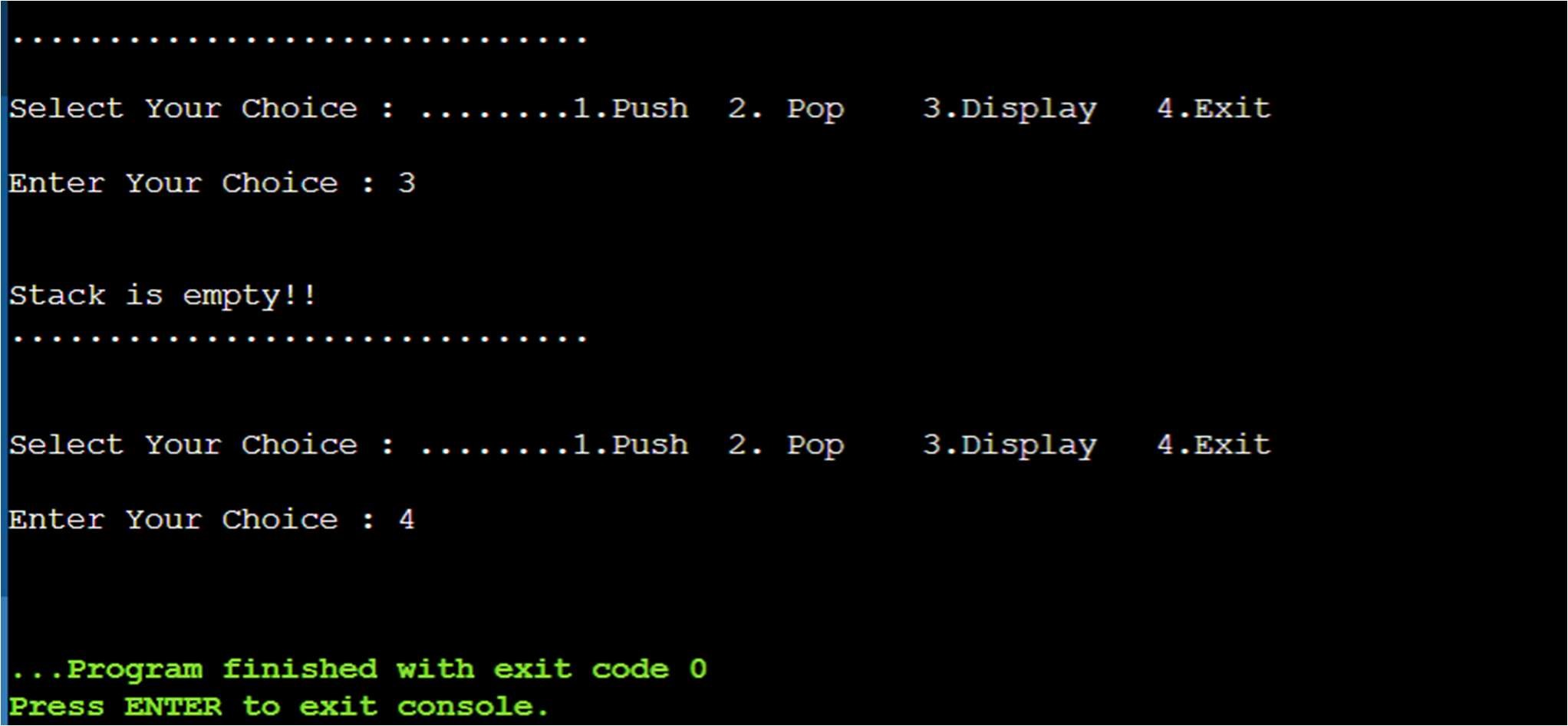
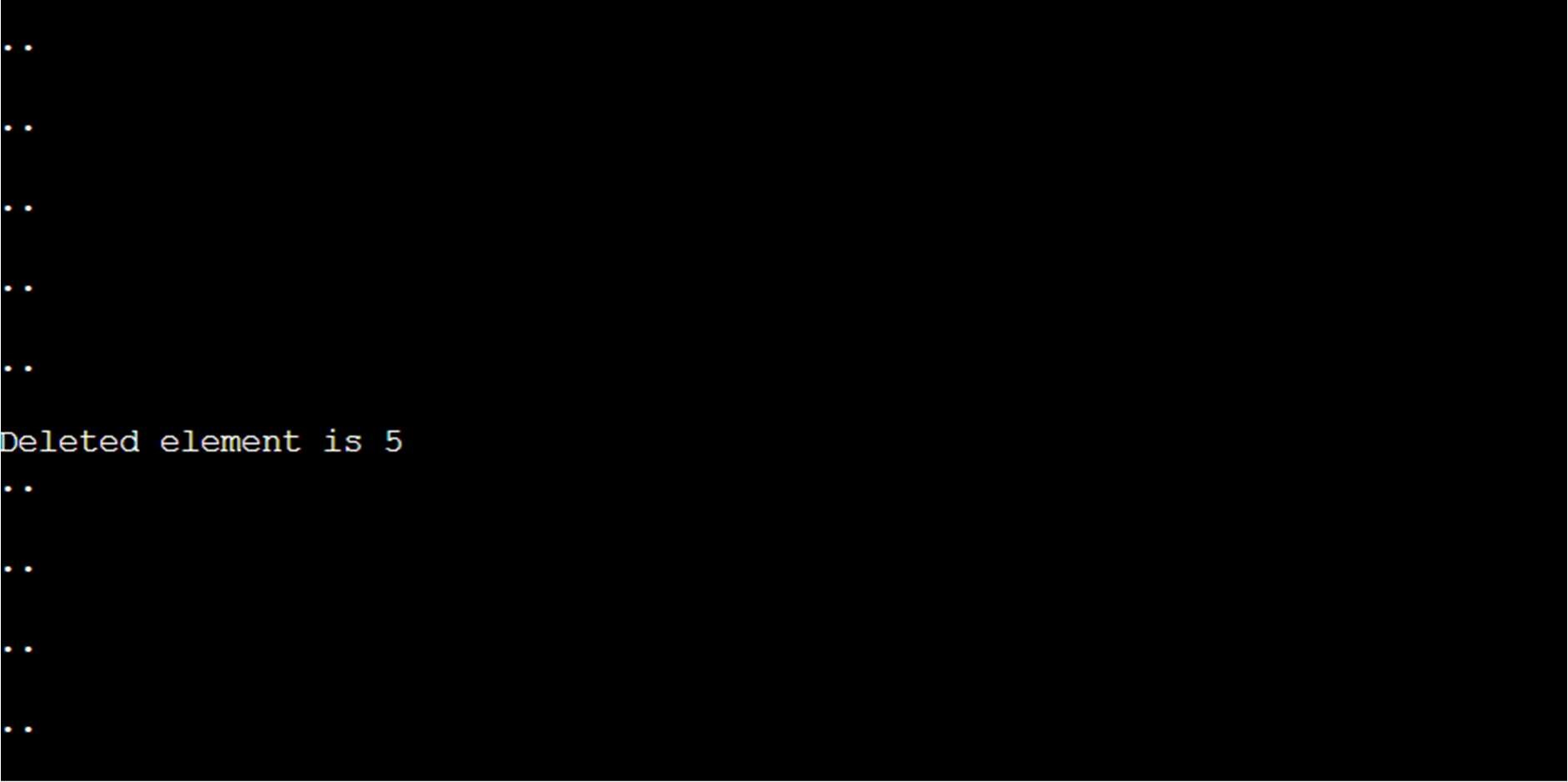
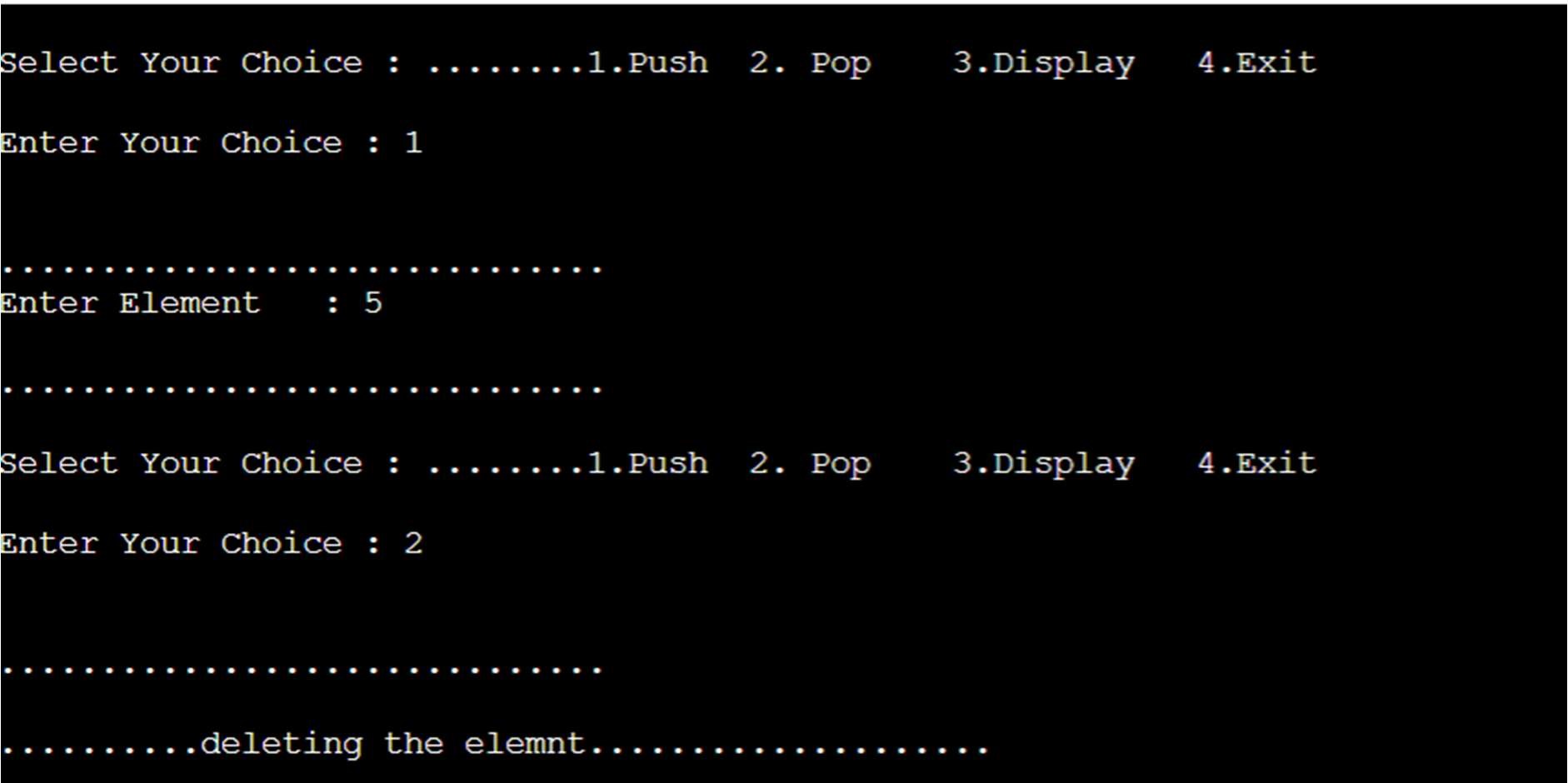
case 2: pop(); break; case 3: display(); break; case 4: exit(0); break; default : cout << "Invalid Choice"; break;

}

}

//return 0;

}



Experiment 6: Program to convert an infix expression to a postfix expression using stacks.

CODE:

#include<stdio.h>

#include<stdlib.h> /\* for exit() \*/

#include<ctype.h> /\* for isdigit(char ) \*/

#include<string.h> #define SIZE 100 char stack[SIZE]; int top = -1; void push(char item){ if(top >= SIZE-1){ printf("\nStack Overflow.");

}

else{ top = top+1; stack[top] = item;

}

}

char pop(){ char item ; if(top <0)

{

printf("stack under flow: invalid infix expression");

getchar();

exit(1);

}

else{ item = stack[top]; top = top-1; return(item);

}

}

int is\_operator(char symbol){

if(symbol == '^' || symbol == '\*' || symbol == '/' || symbol == '+' || symbol =='-'){ return 1;

}

else{ return 0;

}

}

int precedence(char symbol){ if(symbol == '^'){ return(3);

}

else if(symbol == '\*' || symbol == '/'){

return(2);

}

else if(symbol == '+' || symbol == '-'){

return(1);

}

else{ return(0);

}

}

void InfixToPostfix(char infix\_exp[], char postfix\_exp[]){ int i, j;

char item; char x; push('('); strcat(infix\_exp,")"); i=0; j=0; item=infix\_exp[i]; while(item != '\0'){ if(item == '('){ push(item);

}

else if( isdigit(item) || isalpha(item)){

postfix\_exp[j] = item; j++;

}

else if(is\_operator(item) == 1){ x=pop();

while(is\_operator(x) == 1 && precedence(x)>= precedence(item)){ postfix\_exp[j] = x; j++; x = pop();

}

push(x); push(item);

}

else if(item == ')'){ x = pop(); while(x != '('){ postfix\_exp[j] = x; j++; x = pop();

}

}

else{

printf("\nInvalid infix Expression.\n");

getchar(); exit(1);

}

i++; item = infix\_exp[i];

}

if(top>0){

printf("\nInvalid infix Expression.\n"); getchar(); exit(1);

}

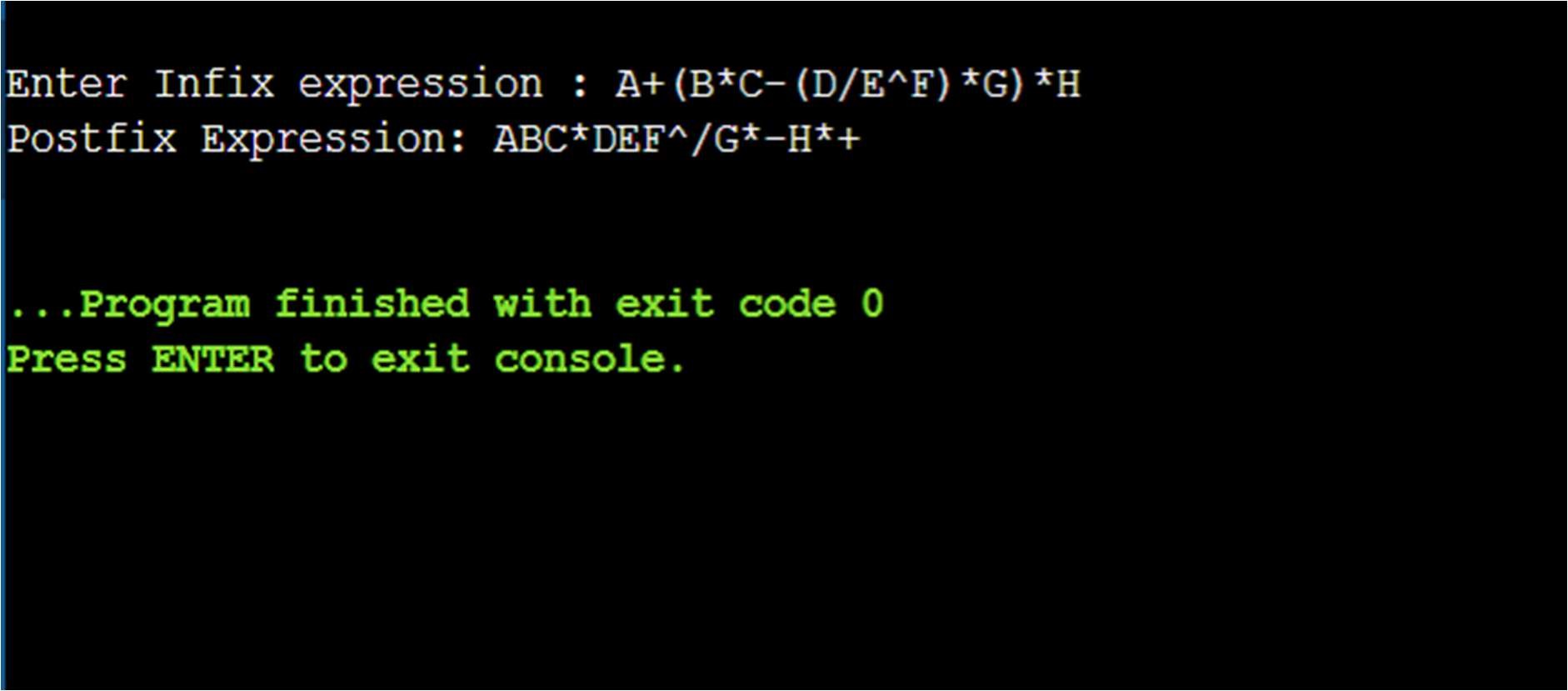
postfix\_exp[j] = '\0';

}

int main(){ char infix[SIZE], postfix[SIZE]; printf("\nEnter Infix expression : "); gets(infix); InfixToPostfix(infix,postfix); printf("Postfix Expression: "); puts(postfix); return 0;

}

OUTPUT:



Experiment 7: Program to evaluate a postfix expression using stacks.

CODE:

#include<stdio.h>

#include<ctype.h>

# define MAXSTACK 100 # define POSTFIXSIZE 100 int stack[MAXSTACK]; int top = -1 ; void push(int item){ if(top >= MAXSTACK -1)

{

printf("stack over flow"); return;

}

else{

top = top + 1 ; stack[top]= item;

}

}

int pop()

{

int item; if(top <0)

{

printf("stack under flow");

}

else{ item = stack[top]; top = top - 1; return item;

}

}

void EvalPostfix(char postfix[])

{ int i ;

char ch; int val; int A, B ; for (i = 0 ; postfix[i] != ')'; i++) {

ch = postfix[i]; if (isdigit(ch))

{

push(ch - '0');

}

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/')

{

A = pop(); B = pop(); switch (ch) { case '\*':

{

val = B\*A; printf("%d \n",val); break;

} case '/':

{

val = B/A; printf("%d \n",val); break;

}

case '+':

{

val = B+A; printf("%d \n",val); break;

} case '-':

{

val = B-A; printf("%d \n",val); break;

} }

push(val);

}

}

printf( " \n Result of expression evaluation : %d \n", pop()) ;

}

int main()

{ int i ;

char postfix[POSTFIXSIZE];

printf("ASSUMPTION: There are only four operators(\*, /, +, -) in an expression and operand is single digit only.\n");

printf( " \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

for (i = 0 ; i <= POSTFIXSIZE - 1 ; i++)

{

scanf("%c", &postfix[i]); if ( postfix[i] == ')' )

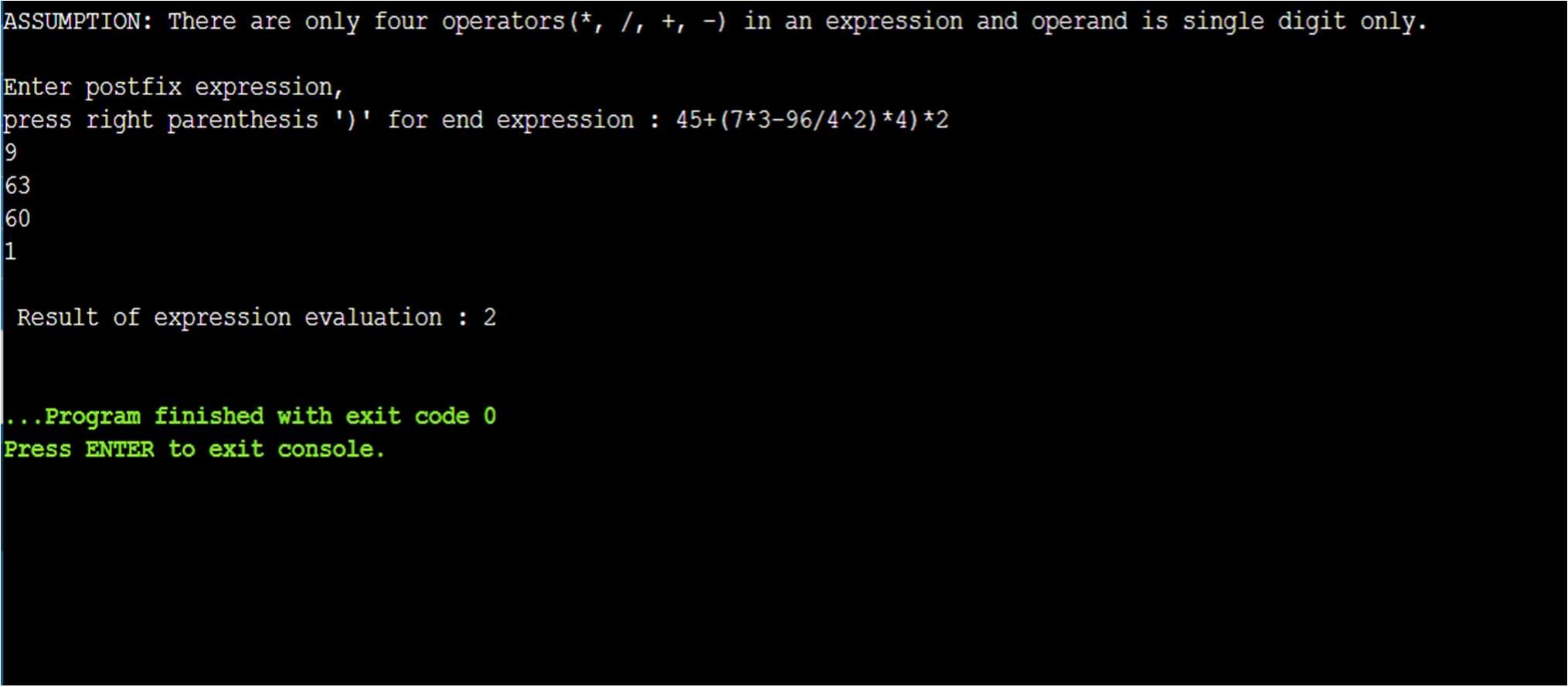
{ break; }

}

EvalPostfix(postfix); return 0;

}

OUTPUT:



Experiment 8: Implement recursive function for Tower of Hanoi problem.

CODE:

#include<iostream> using namespace std;

void TOH(int n,char Beg, char Aux,char End)

{

if(n==1){

cout<<"Move Disk "<<n<<" from "<<Beg<<" to "<<End<<endl; return;

}

else{

TOH(n-1,Beg,End,Aux);

cout<<"Move Disk "<<n<<" from "<<Beg<<" to "<<End<<endl;

TOH(n-1,Aux,Beg,End);

}

}

int main(){

int n;

cout<<"Enter no. of disks:"; cin>>n;

TOH(n,'A','B','C');

return 0;

}

OUTPUT:

