1. **Write a C program to perform Matrix Multiplication**

Input:-

#include<stdio.h>

int main() {

int a[10][10], b[10][10], c[10][10], n, i, j, k;

printf("Enter the value of N (N <= 10): ");

scanf("%d", & n);

printf("Enter the elements of Matrix-A: \n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

scanf("%d", & a[i][j]);

}

}

printf("Enter the elements of Matrix-B: \n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

scanf("%d", & b[i][j]);

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

c[i][j] = 0;

for (k = 0; k < n; k++) {

c[i][j] += a[i][k] \* b[k][j];

}

}

}

printf("The product of the two matrices is: \n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

printf("%d\t", c[i][j]);

}

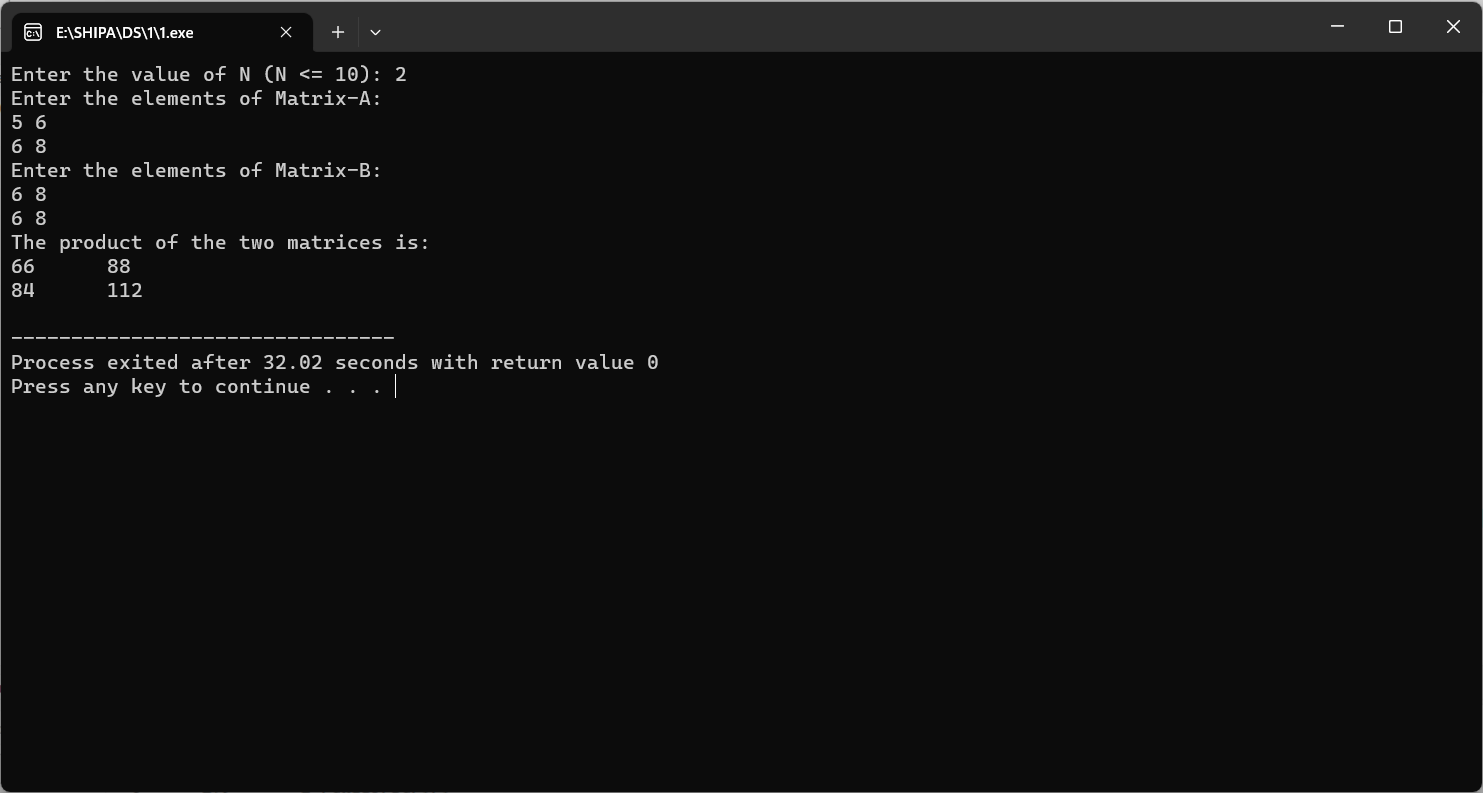
printf("\n");

}

return 0;

}

Output:-



1. **Write a C program to find Odd or Even number from a given set of numbers**

Input:

#include <stdio.h>

int main() {

int num[10],i;

printf("Enter 10 numbers: ");

for(i=0;i<10;i++)

scanf("%d", &num[i]);

printf("\nEven numbers are:\n");

for(i=0;i<10;i++)

{

if(num[i] % 2 == 0)

printf("%d ", num[i]);

}

printf("\nOdd numbers are:\n");

for(i=0;i<10;i++)

{

if(num[i] % 2 != 0)

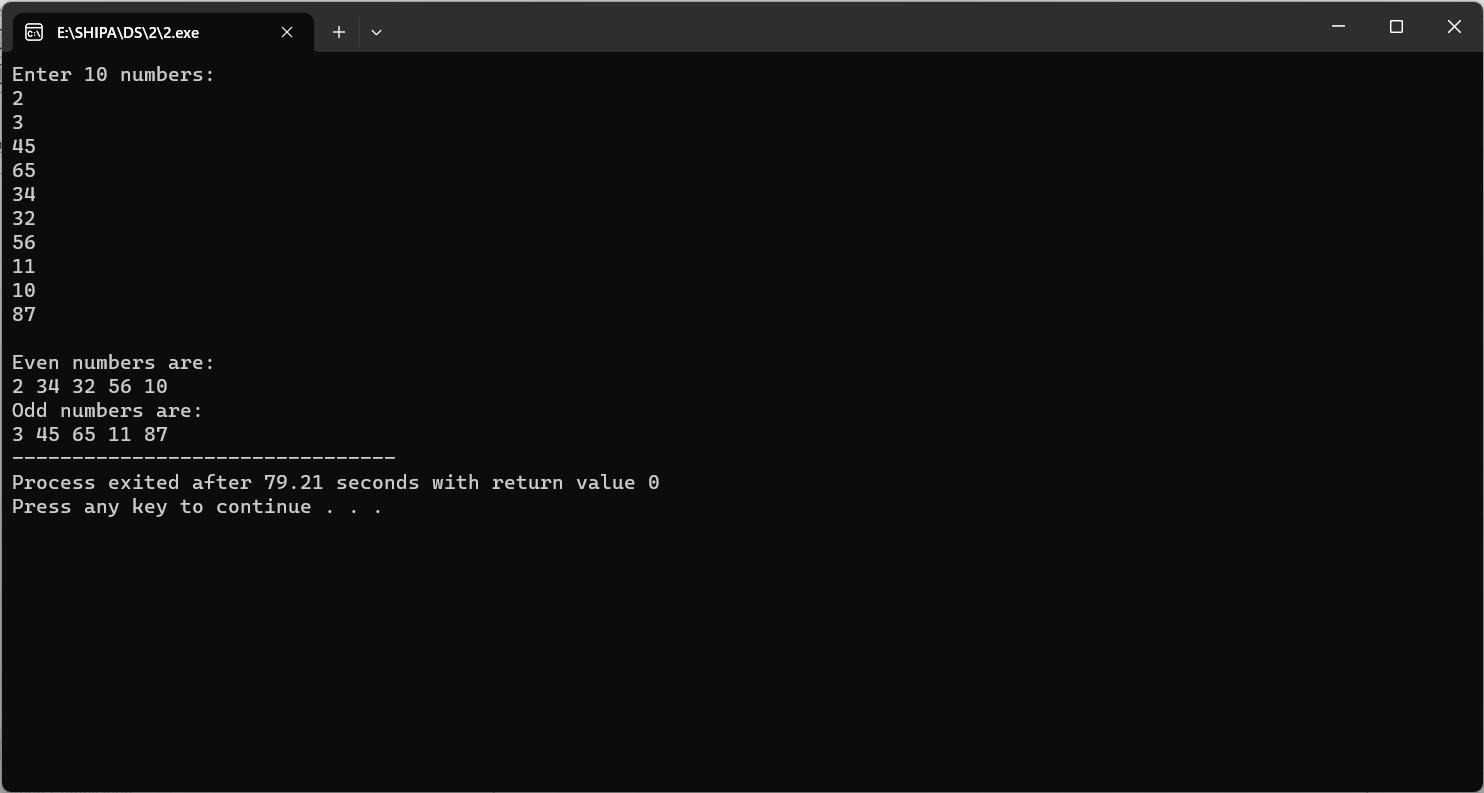
printf("%d ", num[i]);

}

return 0;

}

Output:-



1. **Write a C program to find Factorial of a given number without using Recursion.**

Input:-

#include <stdio.h>

#include<conio.h>

int factorial(int);

int main()

{

int n,fact;

printf("Enter a positive integer: ");

scanf("%d", &n);

fact= factorial(n);

printf("Factorial of %d = %d",n, fact) ;

getch();

}

int factorial(int num)

{

int i=1,f=1;

while(i<=num)

{

f=f\*i;

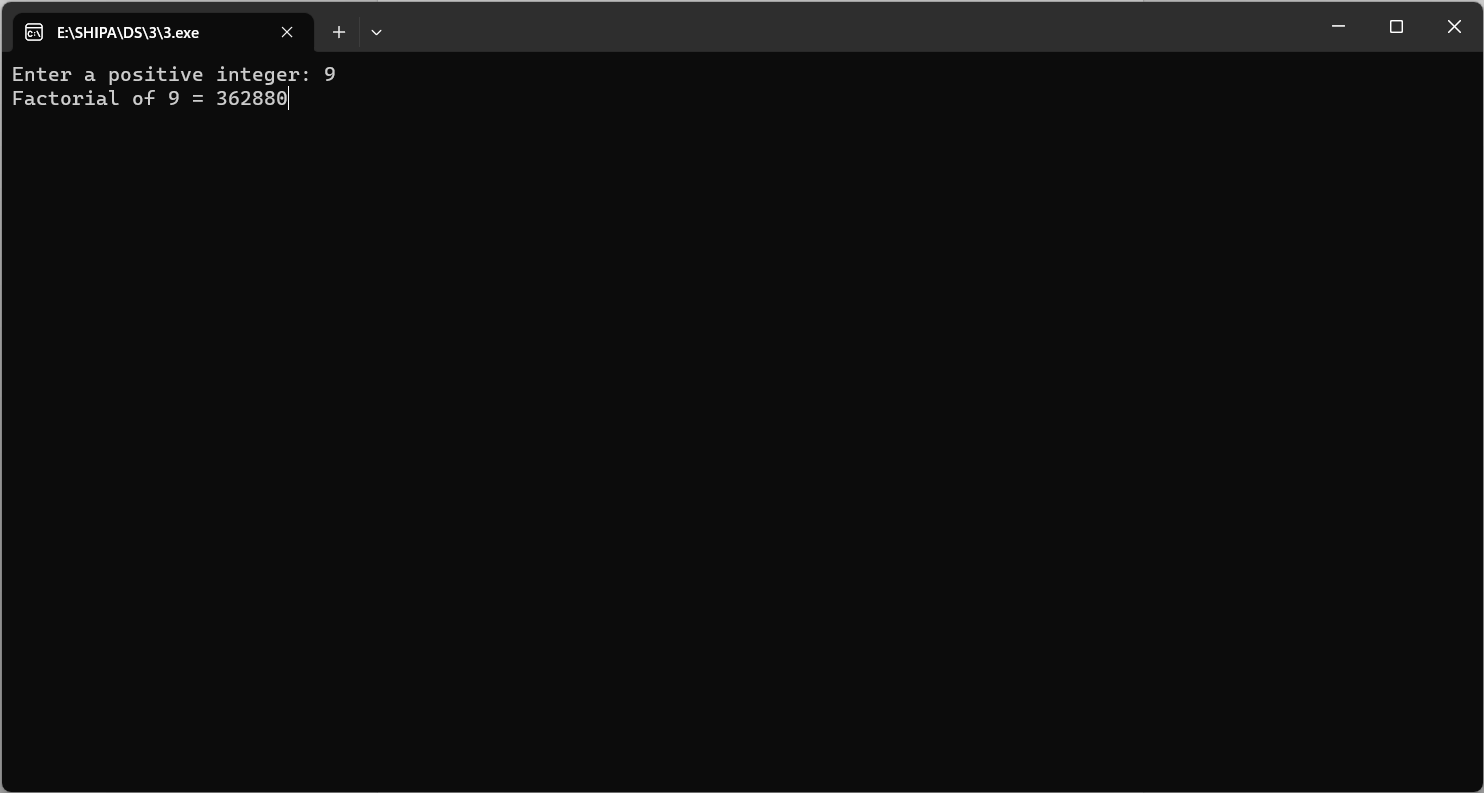
i++;

}

return f;

}

Output:



1. **Write a C program to find Fibonacci series without using Recursion**

Input:

#include <stdio.h>

void fibonacciSeries(int n) {

int first = 0, second = 1, next;

printf("Fibonacci Series up to %d terms:\n", n);

for (int i = 0; i < n; i++) {

printf("%d, ", first);

next = first + second;

first = second;

second = next;

}

}

int main() {

int terms;

printf("Enter the number of terms: ");

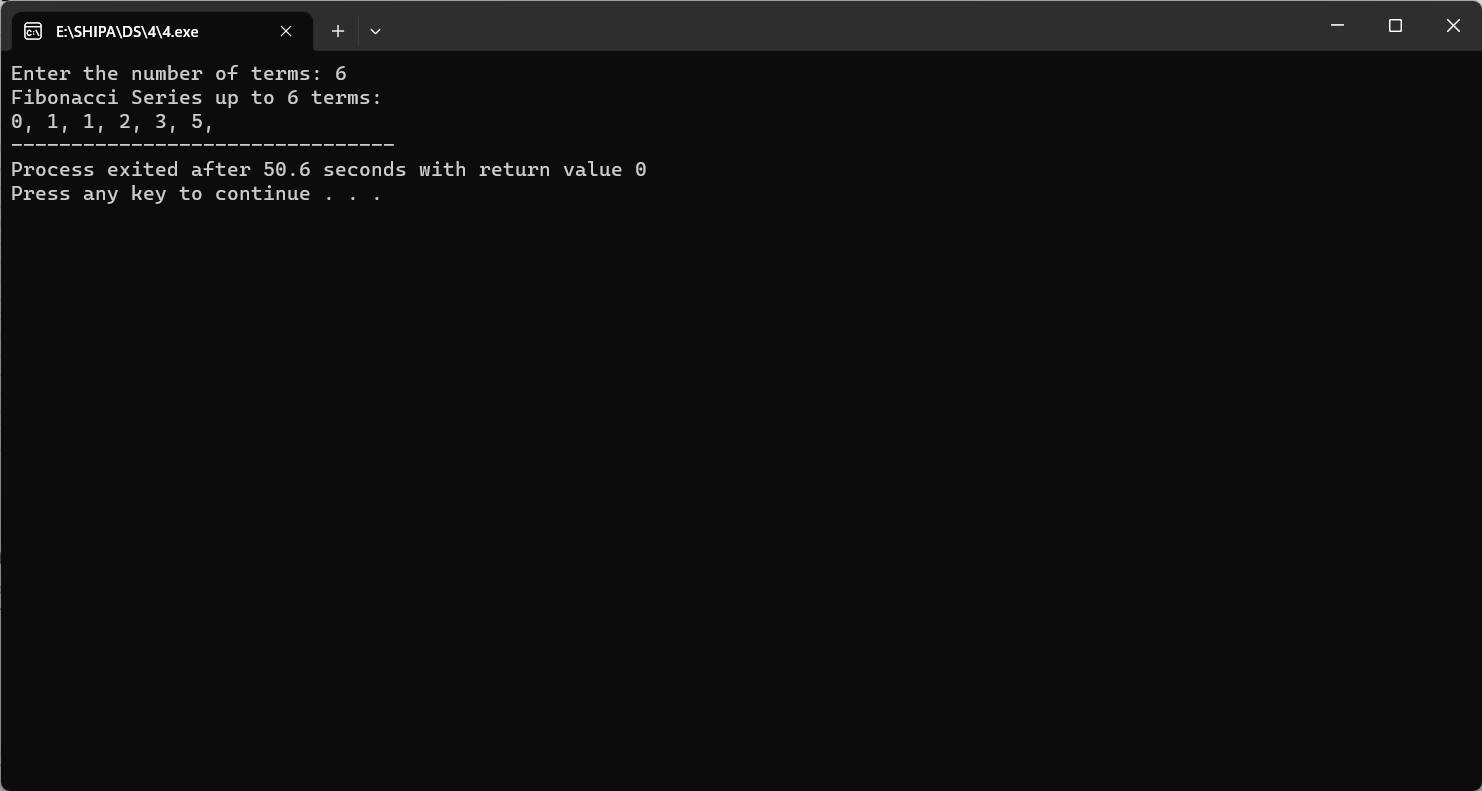
scanf("%d", &terms);

fibonacciSeries(terms);

return 0;

}

Output:



1. **Write a C program to find Factorial of a given number using Recursion**

Input:

#include <stdio.h>

int factorial(int num);

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d", &n);

if (n < 0) {

printf("Factorial is not defined for negative numbers.\n");

} else {

int fact = factorial(n);

printf("Factorial of %d = %d\n", n, fact);

}

return 0;

}

int factorial(int num) {

if (num == 0 || num == 1) {

return 1;

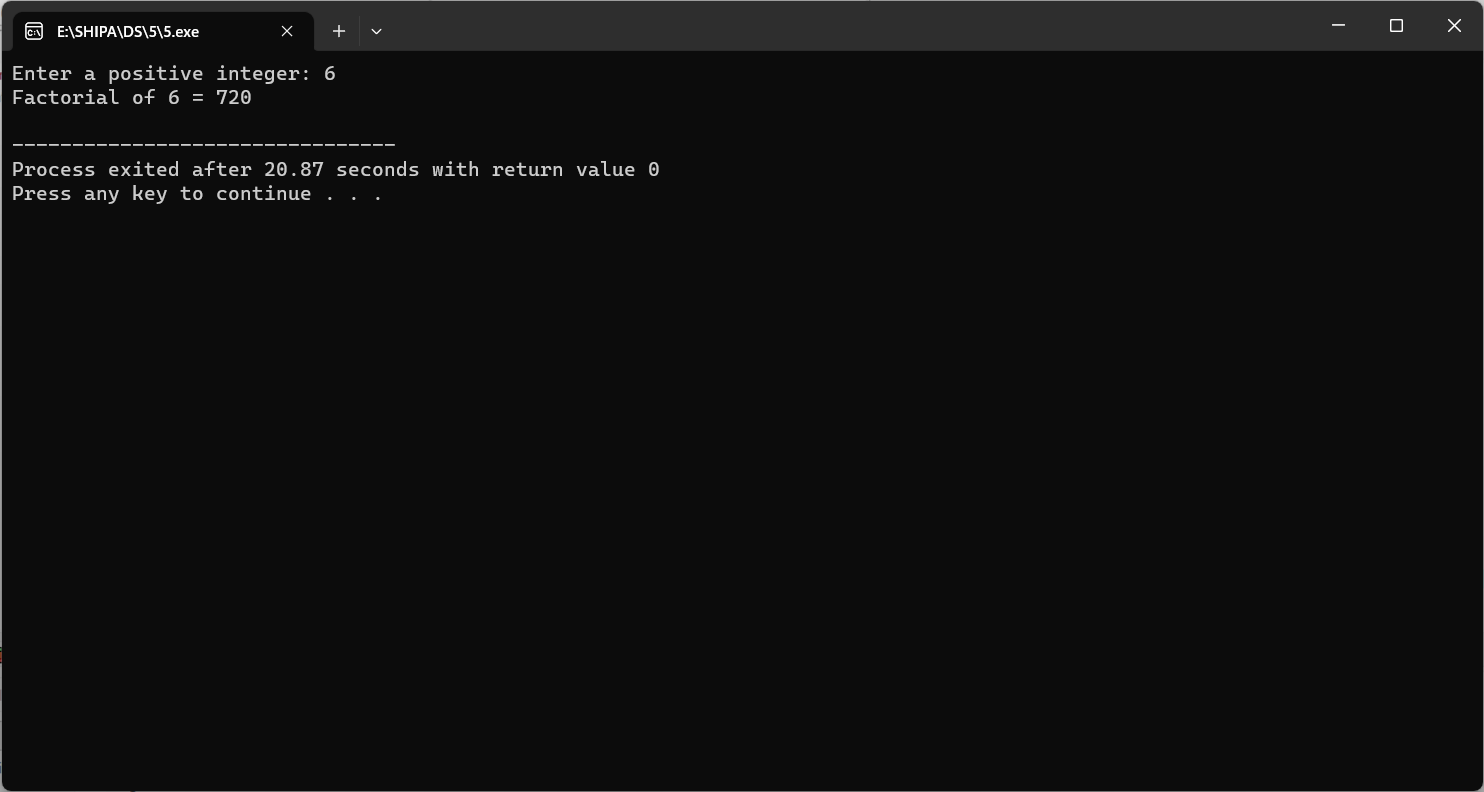
} else {

return num \* factorial(num - 1);

}

}

Output:



1. **Write a C program to find Fibonacci series using Recursion**

Input:

#include<stdio.h>

//Function Definition

void my\_fibonacci(int n)

{

static int n1=0,n2=1,n3;

if(n>0)

{

n3 = n1 + n2;

n1 = n2;

n2 = n3;

printf("%d ",n3);

my\_fibonacci(n-1);

}

}

int main(){

int n;

printf("Number of elements: ");

scanf("%d",&n);

printf("Fibonacci Series: \n");

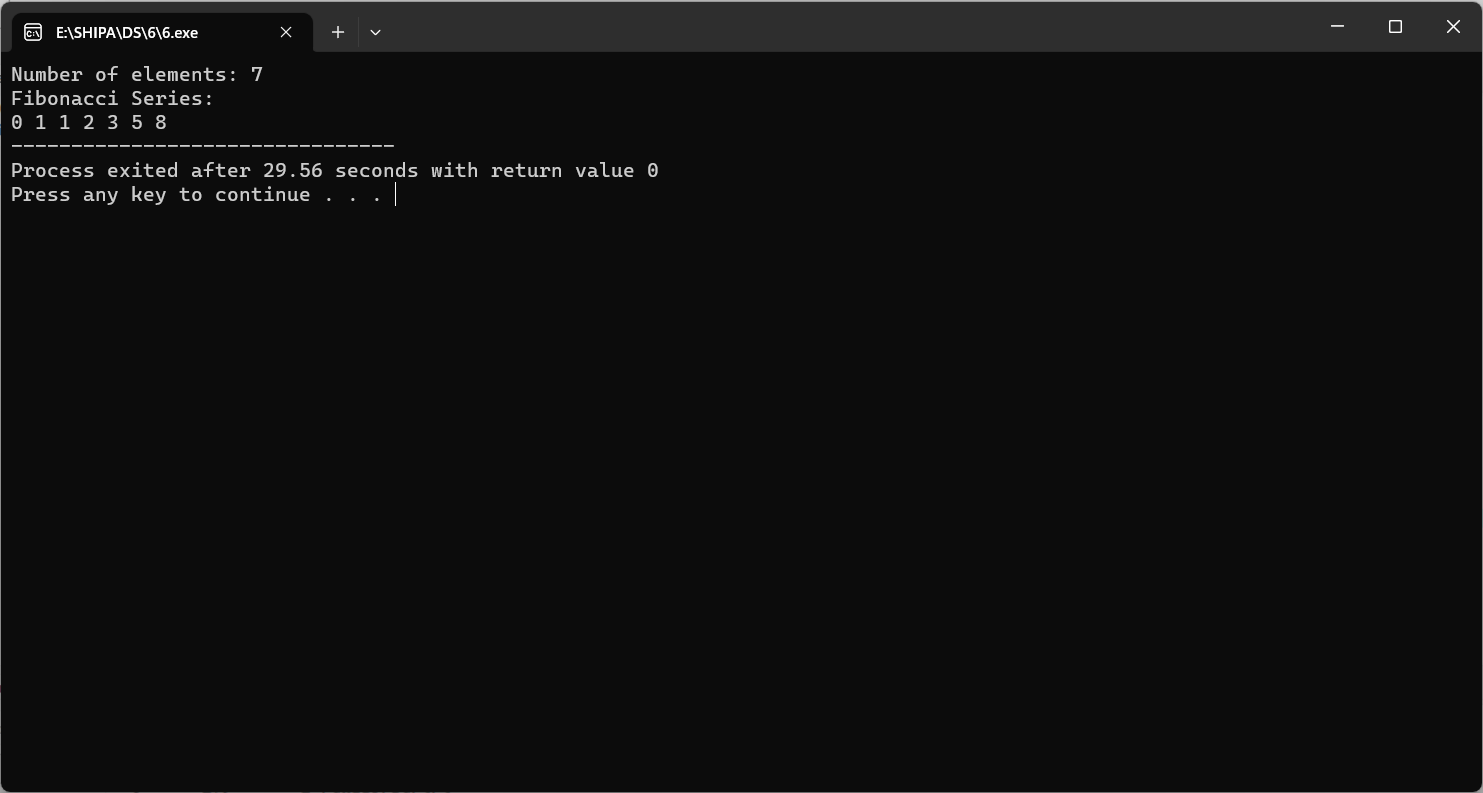
printf("%d %d ",0,1);

my\_fibonacci(n-2);

return 0;

}

Output:



1. **Write a C program to implement Array operations such as Insert, Delete and Display**

Input:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#define MAX 10

void create();

void insert();

void deletion();

void search();

void display();

int a,b[20], n, p, e, f, i, pos;

int main()

{

int ch;

char g='y';

do

{

printf("\n main Menu");

printf("\n 1.Create \n 2.Delete \n 3.Search \n 4.Insert \n 5.Display\n 6.Exit \n");

printf("\n Enter your Choice:");

scanf("%d", &ch);

switch(ch)

{

case 1:

create();

break;

case 2:

deletion();

break;

case 3:

search();

break;

case 4:

insert();

break;

case 5:

display();

break;

case 6:

exit(0);

break;

default:

printf("\n Enter the correct choice:");

}

printf("\n Do u want to continue:::");

scanf("\n%c", &g);

}

while(g=='y'||g=='Y');

getch();

}

void create()

{

printf("\n Enter the number of nodes:");

scanf("%d", &n);

for(i=0;i<n;i++)

{

printf("\n Enter the Element:%d:",(i+1));

scanf("%d", &b[i]);

}

}

void deletion()

{

printf("\n Enter the position u want to delete::");

scanf("%d", &pos);

if(pos>=n)

{

printf("\n Invalid Location::");

}

else

{

for(i=pos+1;i<n;i++)

{

b[i-1]=b[i];

}

n--;

}

printf("\n The Elements after deletion:");

for(i=0;i<n;i++)

{

printf("\t%d", b[i]);

}

}

void search()

{

printf("\n Enter the Element to be searched:");

scanf("%d", &e);

int pos,flag=0;

for(i=0;i<n;i++)

{

if(b[i]==e)

{

flag=1;

pos=i;

break;

}

}

if(flag==1)

{

printf("Value is in the %d Position", pos+1);

}

else

{

printf("Value not found in the list");

}

}

void insert()

{

printf("\n Enter the position u need to insert::");

scanf("%d", &pos);

if(pos>=n)

{

printf("\n invalid Location::");

}

else

{

for(i=MAX-1;i>=pos-1;i--)

{

b[i+1]=b[i];

}

printf("\n Enter the element to insert::\n");

scanf("%d",&p);

b[pos]=p;

n++;

}

printf("\n The list after insertion::\n");

display();

}

void display()

{

printf("\n The Elements of The list ADT are:");

for(i=0;i<n;i++)

{

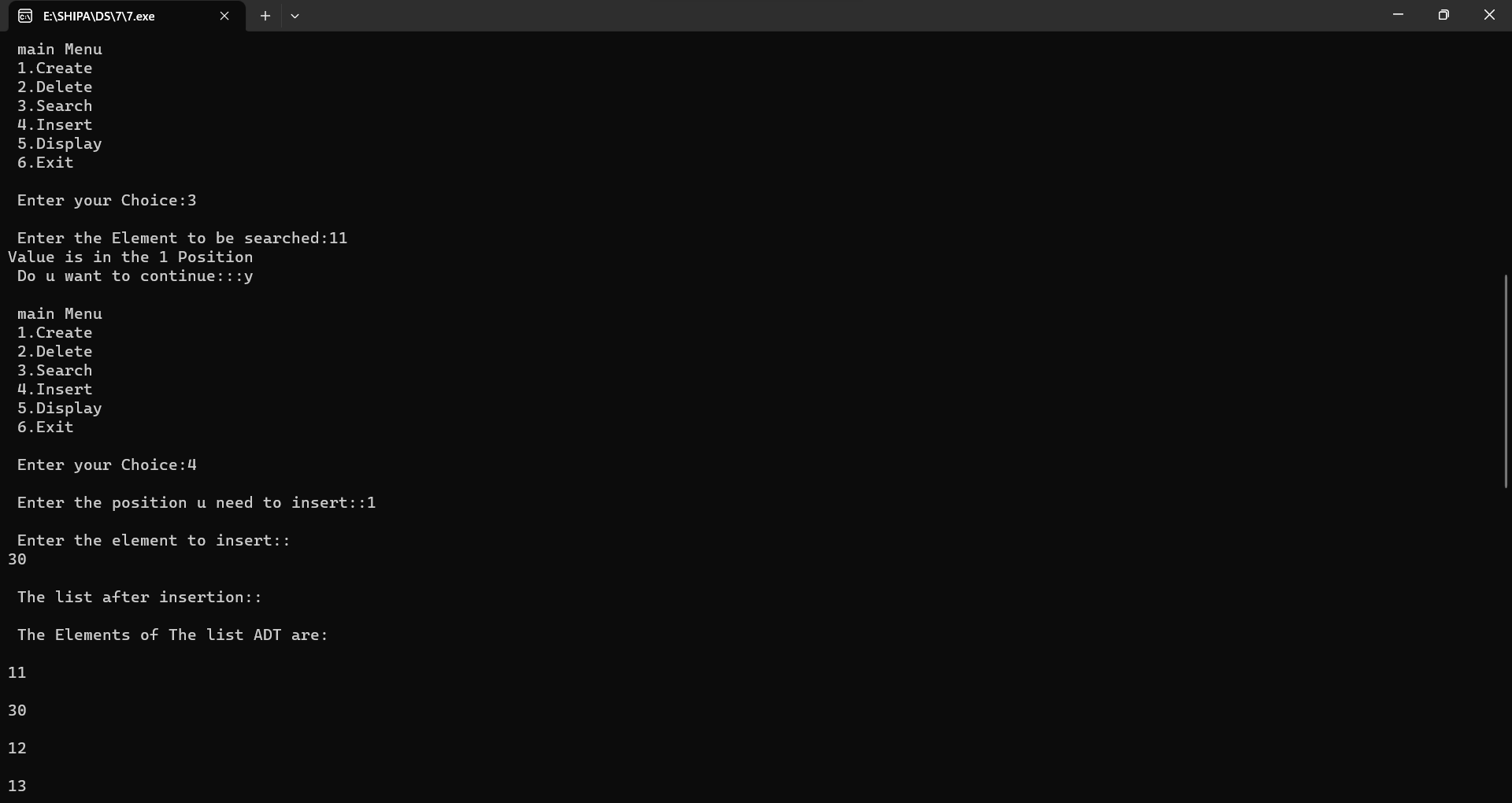
printf("\n\n%d", b[i]);

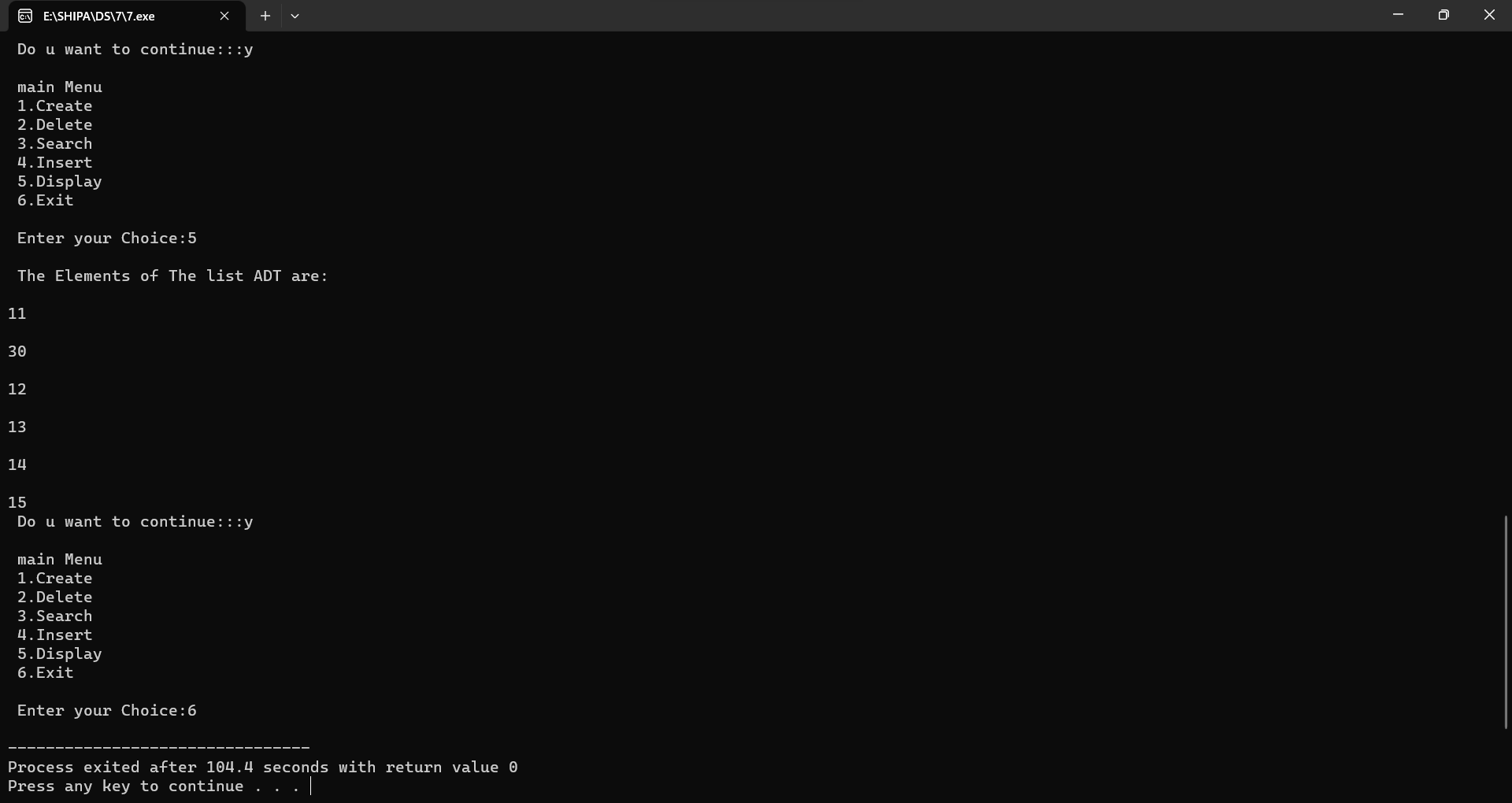
}

}

Output:



****

****

1. **Write a C program to search a number using Linear Search method**

Input:

#include<stdio.h>

int linear(int [],int );

int main(){

int keyElement,i;

int arr[10];

printf("Enter 10 elements:");

for(i=0;i<10;i++)

{

scanf("%d",&arr[i]);

}

printf("Enter the element to search: ");

scanf("%d", &keyElement);

if(linear(arr,keyElement))

printf("The element is found\n");

else

printf("The element is not found\n");

}

int linear(int arr[],int keyElement){

for(int i=0;i<7;i++){

if(arr[i]==keyElement){

return 1;

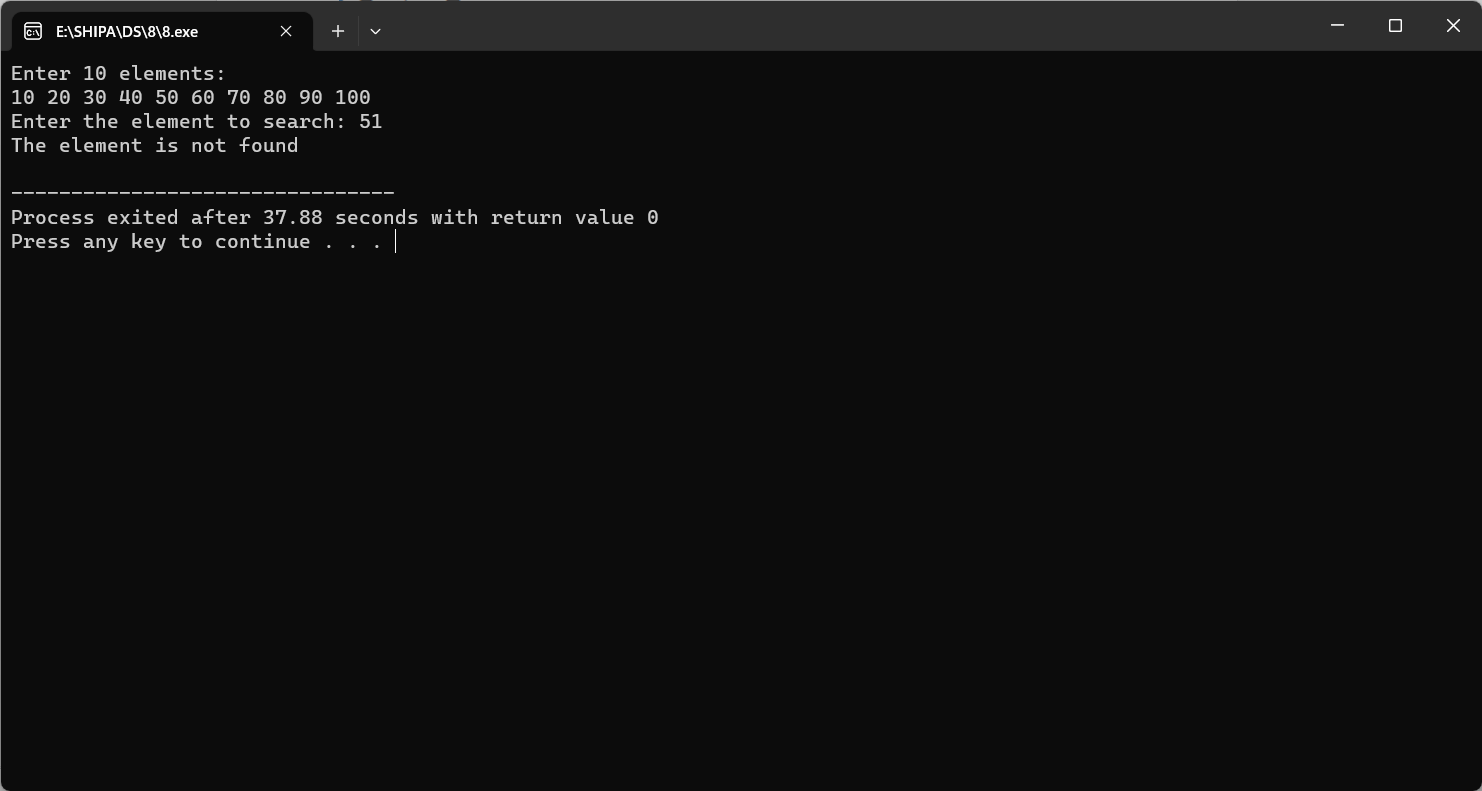
}

}

return 0;

}

Output:



1. **Write a C program to search a number using Binary Search method**

Input:

#include <stdio.h>

int binarySearch(int a[], int beg, int end, int val)

{

int mid;

if(end >= beg)

{ mid = (beg + end)/2;

if(a[mid] == val)

{

return mid+1;

}

/\* if the item to be searched is smaller than middle, then it can only be in left subarray \*/

else if(a[mid] < val)

{

return binarySearch(a, mid+1, end, val);

}

/\* if the item to be searched is greater than middle, then it can only be in right subarray \*/

else

{

return binarySearch(a, beg, mid-1, val);

}

}

return -1;

}

int main() {

int a[] = {11, 14, 25, 30, 40, 41, 52, 57, 70}; // given array

int val = 40; // value to be searched

int n = sizeof(a) / sizeof(a[0]); // size of array

int res = binarySearch(a, 0, n-1, val); // Store result

printf("The elements of the array are - ");

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

printf("%d ", a[i]);

printf("\nElement to be searched is - %d", val);

if (res == -1)

printf("\nElement is not present in the array");

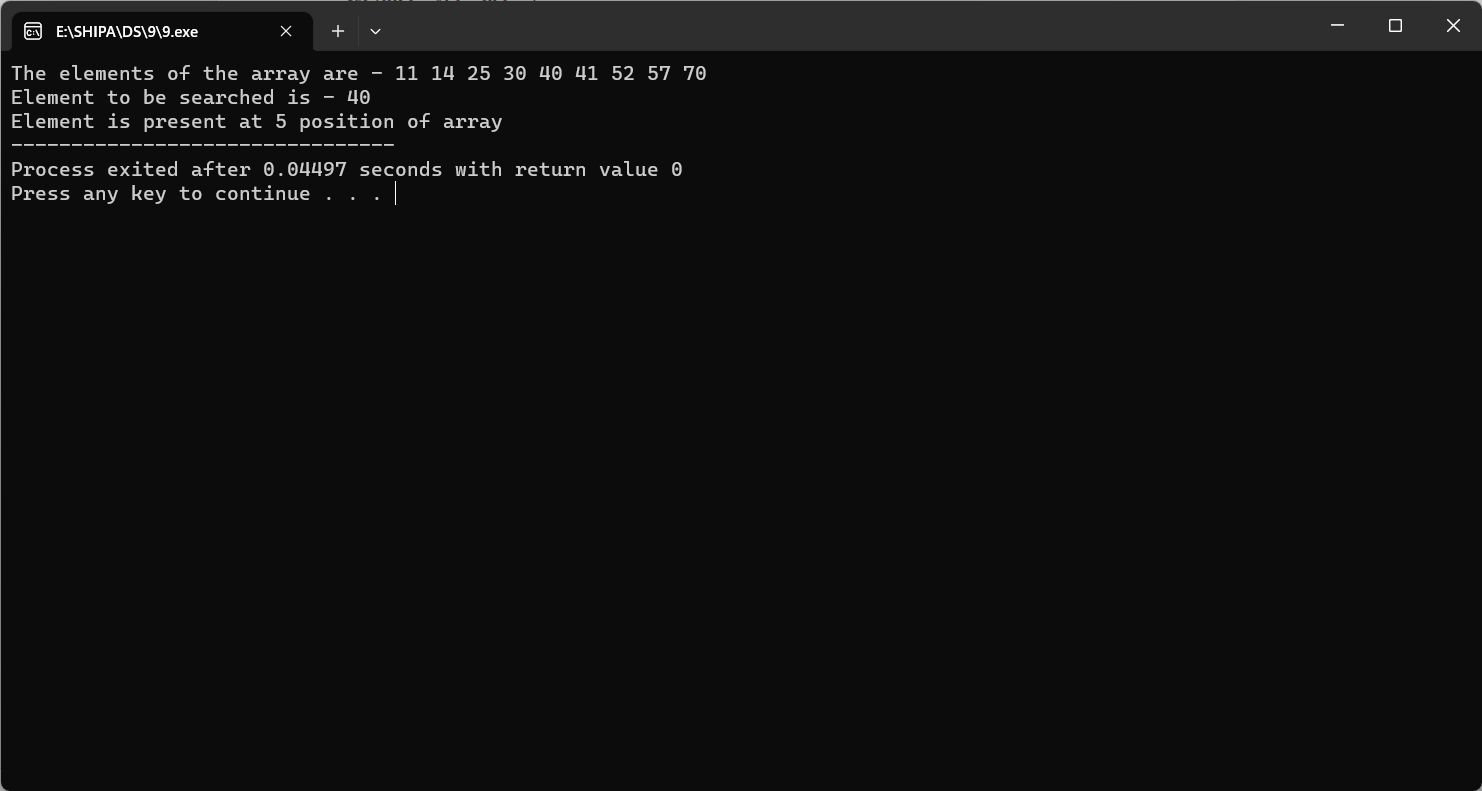
else

printf("\nElement is present at %d position of array", res);

return 0;

}

Output:



1. **Write a C program to implement Linked list operations**

Input:

#include <stdio.h>

#include <stdlib.h>

// Define the singly linked list node

struct Node {

int data;

struct Node \*next;

};

// Function to insert a new node at the beginning of the list

struct Node \*insertAtBeginning(struct Node \*head, int value) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = head;

return newNode;

}

// Function to insert a new node at the end of the list

struct Node \*insertAtEnd(struct Node \*head, int value) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = NULL;

if (head == NULL) {

return newNode;

}

struct Node \*current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

return head;

}

// Function to delete the first occurrence of a node with a given value

struct Node \*deleteNode(struct Node \*head, int value) {

if (head == NULL) {

return NULL;

}

if (head->data == value) {

struct Node \*temp = head;

head = head->next;

free(temp);

return head;

}

struct Node \*current = head;

while (current->next != NULL && current->next->data != value) {

current = current->next;

}

if (current->next != NULL) {

struct Node \*temp = current->next;

current->next = current->next->next;

free(temp);

}

return head;

}

// Function to display the linked list

void display(struct Node \*head) {

struct Node \*current = head;

while (current != NULL) {

printf("%d -> ", current->data);

current = current->next;

}

printf("NULL\n");

}

int main() {

struct Node \*head = NULL;

head = insertAtBeginning(head, 200);

head = insertAtBeginning(head, 10);

head = insertAtEnd(head, 305);

display(head);

head = deleteNode(head, 10);

display(head);

return 0;

}

Output:

