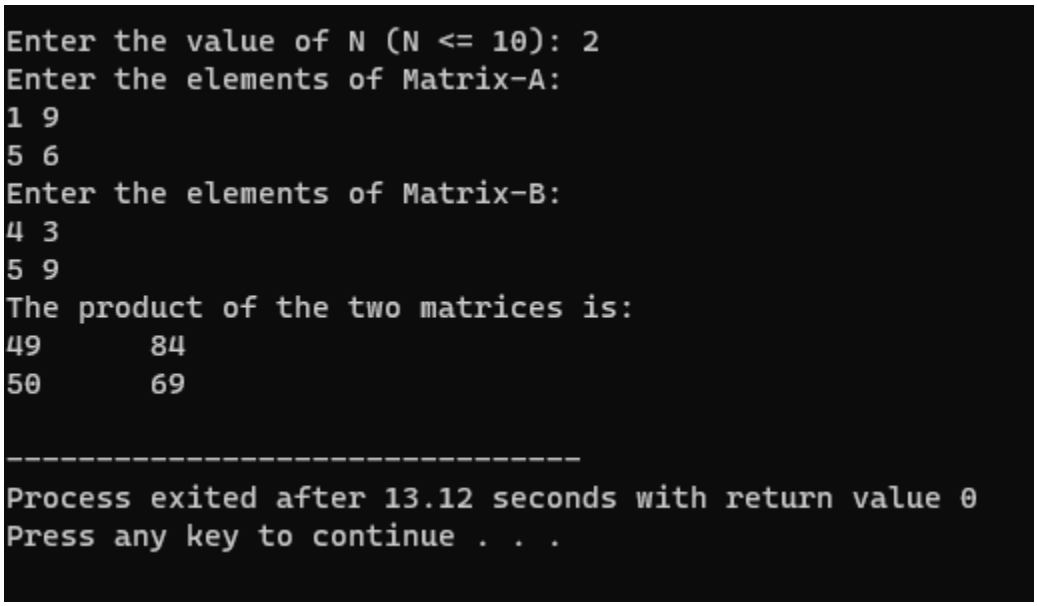
Practical programs (1-10)

1: write a c program to perform matrix multiplication

```
#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;
}
```

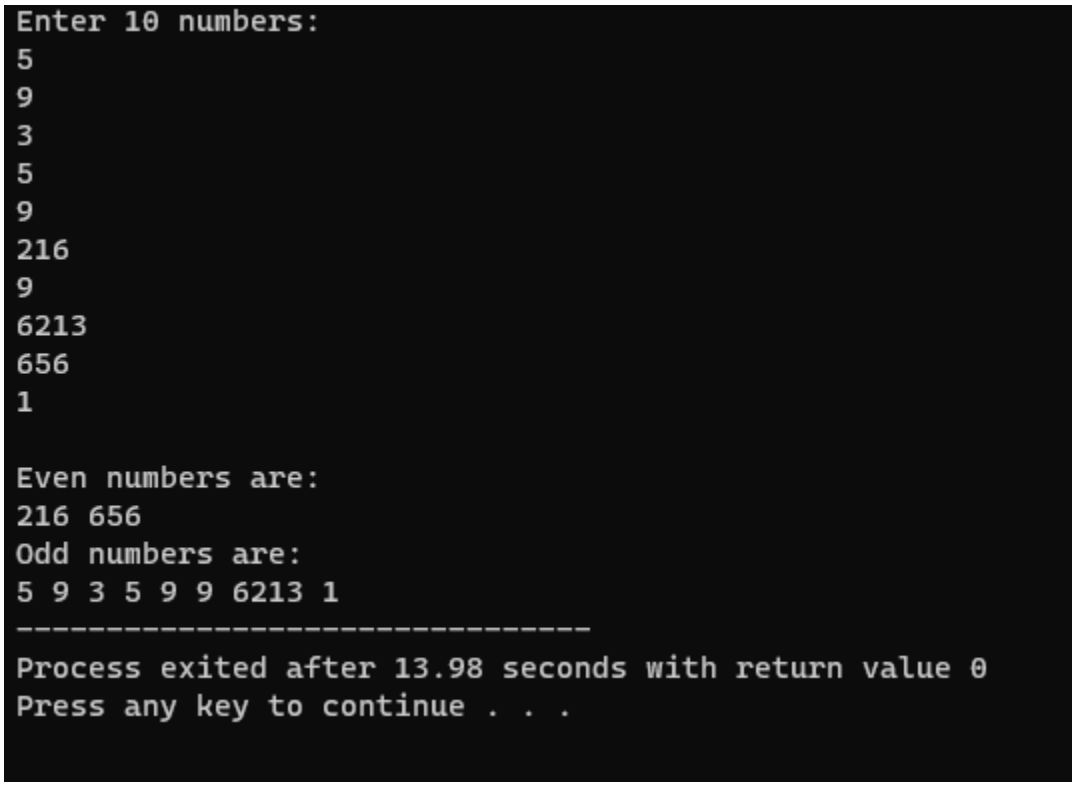
Output:



2. Write a c program to find odd or even numbers from a given set of numbers

```
#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;
}
```

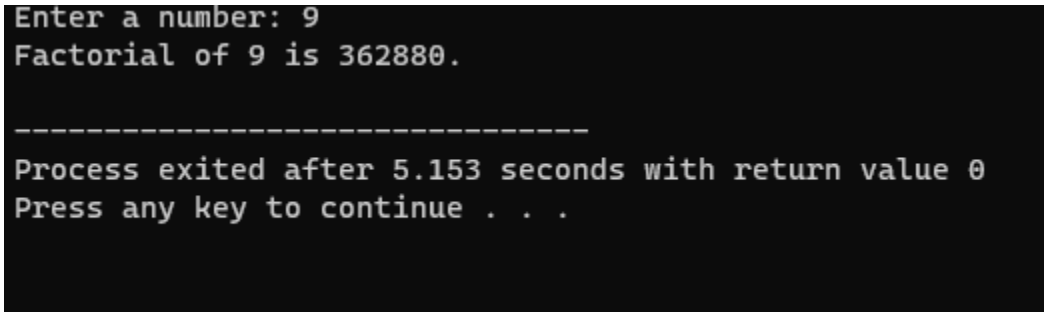
Output:



3. Write a c program to find factorial of a given number without using recursion

```
#include <stdio.h>
int main() {
    int num, factorial = 1;
    printf("Enter a number: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else if (num == 0 || num == 1) {
        printf("Factorial of %d is 1.\n", num);
    } else {
        for (int i = 2; i <= num; i++) {
            factorial *= i;
        }
        printf("Factorial of %d is %d.\n", num, factorial);
    }
    return 0;
}
```

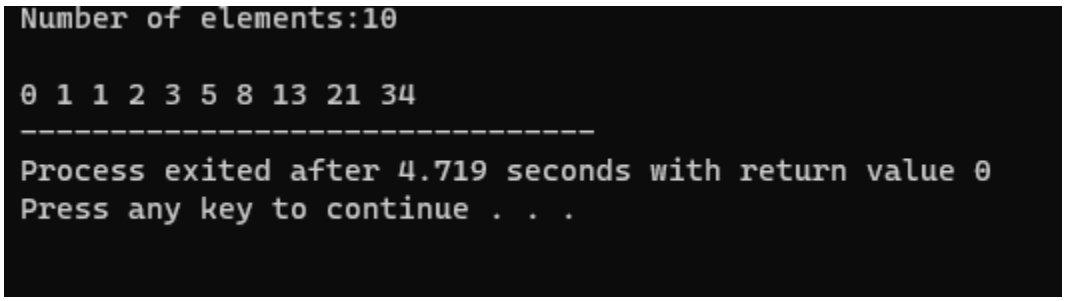
Output:



4. Write a c program to find fibonacci series without using recursion

```
#include<stdio.h>
int main()
{
int n1=0,n2=1,n3,i,num;
printf("Number of elements:");
scanf("%d",&num);
//To print first 0, and 1.
printf("\n%d %d",n1,n2);
for(i=2; i < num; ++i)
{
n3=n1+n2;
printf(" %d",n3);
n1=n2;
n2=n3;
}
return 0;
}
```

Output:



5. Write a c program to find a factorial of a given number using recursion

```
#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) ;
}
int factorial(int n)
{
if (n==1) //Base case
return 1;
else
return n*factorial(n-1); // Inductive step
}
```

Output:

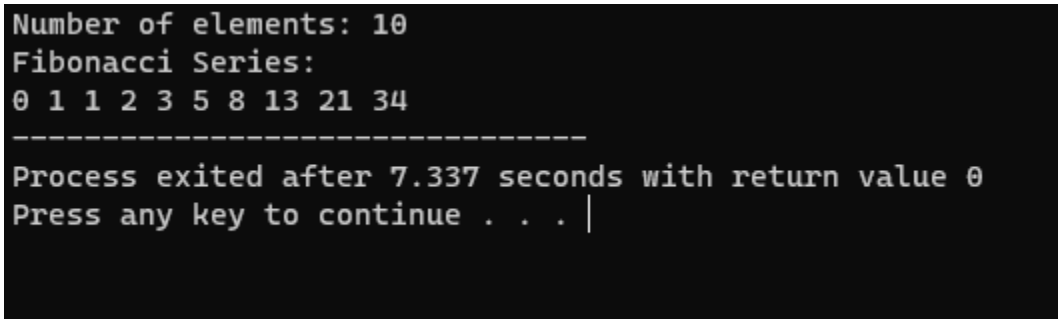
```
Enter a positive integer: 20
Factorial of 20 = -2102132736
--------------------------------
Process exited after 3.685 seconds with return value 0
Press any key to continue . . .
```

6.Write a c program to find fibonacci series using recursion

```
#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:



7.write a c program to implement array operations such as insert,delete,and display

```
#include <stdio.h>
#define MAX_SIZE 100

void insertElement(int arr[], int *size, int position, int element) {
    if (*size >= MAX_SIZE) {
        printf("Array is full. Cannot insert element.\n");
        return;
    }

    if (position < 0 || position > *size) {
        printf("Invalid position for insertion.\n");
        return;
    }

    for (int i = *size; i > position; i--) {
```

```
        arr[i] = arr[i - 1]; // Shift elements to the right
    }

    arr[position] = element;
    (*size)++;
}


void deleteElement(int arr[], int *size, int position) {
    if (*size <= 0 || position < 0 || position >= *size) {
        printf("Invalid position for deletion.\n");
        return;
    }

    for (int i = position; i < *size - 1; i++) {
        arr[i] = arr[i + 1]; // Shift elements to the left
    }

    (*size)--;
}

void displayArray(int arr[], int size) {
    if (size <= 0) {
        printf("Array is empty.\n");
        return;
    }

    printf("Array elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[MAX_SIZE];
    int size = 0;

    int choice, position, element;

    while (1) {
        printf("\nArray Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
```

```
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter position and element to insert: ");
                scanf("%d %d", &position, &element);
                insertElement(arr, &size, position, element);
                Break;

            case 2:
                printf("Enter position to delete: ");
                scanf("%d", &position);
                deleteElement(arr, &size, position);
                Break;

            case 3:
                displayArray(arr, size);
                Break;

            case 4:
                printf("Exiting program.\n");
                return 0;

            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

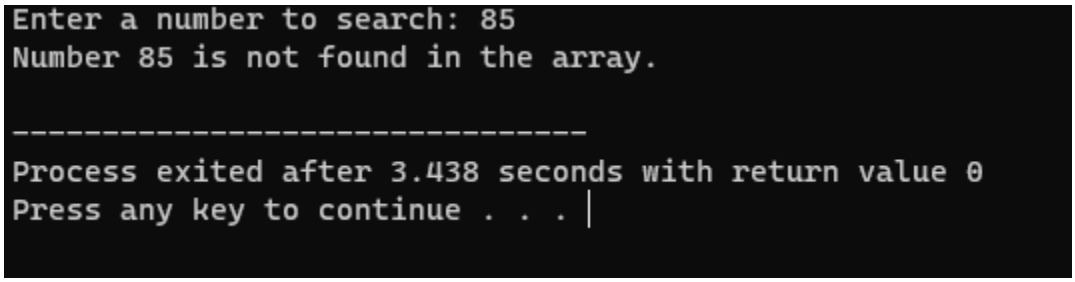
Output:

```
Array Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter position and element to insert: 0 10

Array Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter position and element to insert: 1 20

Array Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2 30
Enter position to delete: Invalid position for deletion.

Array Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Array elements: 10 20

Array Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter position and element to insert: 0 30

Array Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Array elements: 30 10 20
```

8. Write a c program to search a number using linear search method

```
#include <stdio.h>
int main() {
    int arr[] = { 10, 5, 8, 20, 13, 7 };
    int size = sizeof(arr) / sizeof(arr[0]);

    int target;
    printf("Enter a number to search: ");
    scanf("%d", &target);

    int found = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            found = 1;
            break;
        }
    }

    if (found) {
        printf("Number %d is found in the array.\n", target);
    } else {
        printf("Number %d is not found in the array.\n", target);
    }

    return 0;
}
```

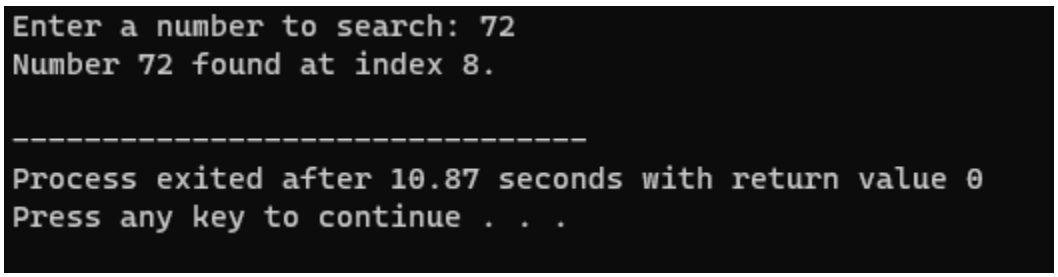
Output:



9. Write a c program to search a number using binary search method

```
#include <stdio.h>

int binarySearch(int arr[], int size, int target) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid; // Target found, return index
        } else if (arr[mid] < target) {
            left = mid + 1; // Search the right half
        } else {
            right = mid - 1; // Search the left half
        }
    }

    return -1; // Target not found
}
int main() {
    int arr[] = { 2, 5, 8, 12, 16, 23, 38, 56, 72, 91 };
    int size = sizeof(arr) / sizeof(arr[0]);

    int target;
    printf("Enter a number to search: ");
    scanf("%d", &target);
    int index = binarySearch(arr, size, target);

    if (index != -1) {
        printf("Number %d found at index %d.\n", target, index);
    } else {
        printf("Number %d not found in the array.\n", target);
    }

    return 0;
}
```

output:



10. Write a c program to implement linked list operation

```
#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:

```
10 -> 200 -> 305 -> NULL
200 -> 305 -> NULL

--------------------------------
Process exited after 2.331 seconds with return value 0
Press any key to continue . . .
```