1Write a C program to implement following operations.

a)traverse

```
#include <stdio.h>
 void printArray(int* arr, int n)
{
	int i;
	printf("Array: ");
	for (i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	printf("\n");
}
int main()
{
	int arr[] = { 2, -1, 5, 6, 0, -3 };
	int n = sizeof(arr) / sizeof(arr[0]);

	printArray(arr, n);

	return 0;
}
```

Output:

Array: 2 -1 5 6 0 -3

b)search

```
#include <stdio.h>
 int linearSearch(int* arr, int size, int key)
{
      for (int i = 0; i < size; i++) {
            if (arr[i] == key) {
                  return i;
            }
      }
      return -1;
}
int main()
{
      int arr[10] = { 3, 4, 1, 7, 5, 8, 11, 42, 3, 13 };
      int size = sizeof(arr) / sizeof(arr[0]);
      int key = 4;
       int index = linearSearch(arr, size, key);
       if (index == -1) {
            printf("The element is not present in the arr.");
      }
      else {
            printf("The element is present at arr[%d].", index);
      }
   return 0;
}
```

Output:

The element is present at arr[1].

c)insert

```
#include <stdio.h>
int main()
{
	int arr[100] = { 0 };
	int i, x, pos, n = 10;
	for (i = 0; i < 10; i++)
		arr[i] = i + 1;
	for (i = 0; i < n; i++)
		printf("%d ", arr[i]);
	printf("\n");
	x = 50;
	 pos = 5;
	 n++;
	for (i = n - 1; i >= pos; i--)
		arr[i] = arr[i - 1];
	 arr[pos - 1] = x;
	 for (i = 0; i < n; i++)
		printf("%d ", arr[i]);
	printf("\n");
   return 0;
}
```

Output:

1 2 3 4 5 6 7 8 9 10

1 2 3 4 50 5 6 7 8 9 10

d)delete

```
#include <stdio.h>
int findElement(int arr[], int n, int key);
int deleteElement(int arr[], int n, int key)
{
	int pos = findElement(arr, n, key);
	if (pos == -1) {
		printf("Element not found");
		return n;
	}
	int i;
	for (i = pos; i < n - 1; i++)
		arr[i] = arr[i + 1];

	return n - 1;
}
int findElement(int arr[], int n, int key)
{
	int i;
	for (i = 0; i < n; i++)
		if (arr[i] == key)
```

```
                return i;

    return -1;
}
int main()
{
    int i;
    int arr[] = { 10, 50, 30, 40, 20 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 30;
    printf("Array before deletion\n");
    for (i = 0; i < n; i++)
        printf("%d  ", arr[i]);
    n = deleteElement(arr, n, key);
    printf("\nArray after deletion\n");
    for (i = 0; i < n; i++)
        printf("%d  ", arr[i]);
    return 0;
}
```

output:

Array before deletion:10,50,30,40,20

Array after deletion:10,50,40,20

e)Update

```
#include<stdio.h>
```

```
int main()
{
   int i,t,a[10],n,m,s,j=0,b[10];
   printf("\nEnter the Limit:");
   scanf("%d",&n);
   printf("\nEnter the Values:");
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("\nGiven values are:");
   for(i=0;i<n;i++)
   {
      printf("a[%d]=%d",i,a[i]);
   }
   printf("\nEnter the position to be update:");
   scanf("%d",&t);
   printf("\nEnter the value to be update:");
   scanf("%d",&s);
   for(i=0;i<n;i++)
   {
      if(i==t)
      {
          a[i]=s;
```

```
        }
    }
    printf("\nUpdated value is:");
    for(i=0;i<n;i++)
    {
        printf("\na[%d]=%d",i,a[i]);
    }
    return 0;
}
```

Output:

```
Enter the limit:5
Enter the values:1
2
3
4
5
Given values are:
a[0]=1
a[1]=2
a[2]=3
a[3]=4
a[4]=5
Enter the position to be update:3
Enter the value to be update:5
```

Inserted value is:

a[0]=1

a[1]=2

a[2]=3

a[3]=5

a[4]=4

a[5]=5

2.writing a recursive function to calculate the factorial to calculate the factorial of a number.

```
#include <stdio.h>

long long factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}

int main() {
    int number = 5;
    printf("Factorial of %d is %lld\n", number,
factorial(number));
    return 0;
}
```

output:factorial of 5 is 120.

3.Write a c program to find duplicate element in an array.

```
#include <stdio.h>
```

```
void findDuplicates(int arr[], int size) {
    int i, j;
    printf("Duplicate elements in the array are: ");
    for (i = 0; i < size - 1; i++) {
        for (j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                printf("%d ", arr[i]);
                break;                    }
        }
    }
}
int main() {
    int arr[] = {1, 2, 3, 4, 2, 3, 5, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    findDuplicates(arr, size);
    return 0;
}
```

output:duplicate elements in the array are:2,3

4.write a c programto find max and min from an array elements.

```
#include <stdio.h>
int main() {
    int n, i;
    int max, min;
```

```
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements of the array: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    max = arr[0];
    min = arr[0];
    for(i = 1; i < n; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
        if(arr[i] < min) {
            min = arr[i];
        }
    }
    printf("Maximum element = %d\n", max);
    printf("Minimum element = %d\n", min);
  return 0;
}
```

output:

enter the numbers of element in array:[1,2,3,4,5,6]

maximum element:6

minimum element:1

5.given a number m.the task is to print the fibonacci series and the sum of the series using recursion.

```
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
int main() {
    int n, i;
    int sum = 0;
    printf("Enter the number of terms: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
        sum += fibonacci(i);
    }
    printf("\nSum of the Fibonacci series: %d\n", sum);
    return 0;
}
```

output:

enter the number of terms:10

0 1 1 2 3 5 8 13 21 34

sum of the fibonacci series:88

6.you are given an array arr in increasing order.find the element x from arr using binary

```
#include <stdio.h>
int binarySearch(int arr[], int size, int x) {
    int left = 0;
    int right = size - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            return mid;
  }
        else if (arr[mid] < x)
{
            left = mid + 1;
        }
        else {
            right = mid - 1;
        }
    }

    return -1;
}
int main() {
```

```
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; // Example sorted
array
    int size = sizeof(arr) / sizeof(arr[0]);
    int x = 5; // Element to search for
    int result = binarySearch(arr, size, x);
    if (result != -1) {
        printf("Element %d found at index %d.\n", x, result);
    } else {
        printf("Element %d not found in the array.\n", x);
    }
    return 0;
}
```

output:element 5 found at index 4.

7.linear search

```
#include <stdio.h>
void main()
{
    int num;
    int i,    key, element_found = 0;
    printf("Enter number of elements you would like to take as
input: ");
    scanf("%d", &num);
    int arr[num];
    printf("\nEnter all the elements of your choice:");
    for (i = 0; i < num; i++)
```

```
    {
        scanf("%d", &arr[i]);
    }
    printf("\nEnter the key element that you would like to be searched: ");
    scanf("%d", &key);
    /*  Linear search starts */
    for (i = 0; i < num ; i++)
    {
        if (key == arr[i] )
        {
            element_found = 1;
            break;
        }
    }
    if (element_found == 1)
        printf("we got the element at index %d",i+1);
    else
        printf("we havent got element at any index in the array\n")
}
```

output:

enter all the elements of your choice:1,2,3,4,5

we got the element at index 2

8.Binary search

```
// Binary Search in C

#include <stdio.h>

int binarySearch(int array[], int x, int low, int high) {
  // Repeat until the pointers low and high meet each other
  while (low <= high) {
    int mid = low + (high - low) / 2;

    if (array[mid] == x)
      return mid;

    if (array[mid] < x)
      low = mid + 1;

    else
      high = mid - 1;
  }

  return -1;
}

int main(void) {
  int array[] = {3, 4, 5, 6, 7, 8, 9};
```

```
    int n = sizeof(array) / sizeof(array[0]);
    int x = 4;
    int result = binarySearch(array, x, 0, n - 1);
    if (result == -1)
        printf("Not found");
    else
        printf("Element is found at index %d", result);
    return 0;
}
```