1.Write a C Program to implement the following operations.

A.Traverse

```
#include <stdio.h>
void insertElement(int arr[], int size, int element, int position) {
    if (position >= size) {
        printf("Invalid position for insertion.\n");
        return;
    }
    for (int i = size - 1; i >= position; i--) {
        arr[i + 1] = arr[i];
    }
    arr[position] = element;
}
int main() {
    int array[100] = {1, 2, 3, 4, 5};
    int size = 5;
    int element = 10;
    int position = 2;
    insertElement(array, size, element, position);
    size++;
    printf("Array after insertion:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    return 0;
}
```

Output:

1 2 3 4 5

B.Insert:

```
#include <stdio.h>

void insertElement(int arr[], int size, int element, int position) {
    if (position >= size) {
        printf("Invalid position for insertion.\n");
        return;
    }

    for (int i = size - 1; i >= position; i--) {
        arr[i + 1] = arr[i];
```

```
    }

    arr[position] = element;
}

int main() {
    int array[100] = {1, 2, 3, 4, 5};
    int size = 5;
    int element = 10;
    int position = 2;

    insertElement(array, size, element, position);
    size++;

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

    return 0;
}
```

Output:

Array after insertion:

1 2 10 3 4 5

C.search

```
#include <stdio.h>

int search(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key) {
            return i;          }
    }
    return -1;
}
int main() {
    int arr[] = {2, 4, 6, 8, 10};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 6;
    int result = search(arr, n, key);

    if (result != -1) {
        printf("Element found at index: %d\n", result);
    } else {
```

```
        printf("Element not found\n");
    }

    return 0;
}
```

Output:
Element found at index: 2

d)delete:

```
#include <stdio.h>

void deleteElement(int arr[], int size, int element) {
    int i, j;
    for (i = 0; i < size; i++) {
        if (arr[i] == element) {
            for (j = i; j < size - 1; j++) {
                arr[j] = arr[j + 1];
            }
            size--;
            break;
        }
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = 5;
    int element = 3;

    deleteElement(arr, size, element);

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

    return 0;
}
```

Output:
Array after deletion: 1 2 4 5 5

e)update:

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

```
int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int index = 2;
    int newValue = 10;
   arr[index] = newValue;
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```
Output:;
1 2 10 4 5

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

```
#include <stdio.h>
int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}

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

Output:
Factorial of 5 is 120

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

```
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3, 4, 2, 5, 6, 3};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Duplicate elements in the array are: ");
    for (int i = 0; i < size; i++) {
```

```
        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                printf("%d ", arr[j]);
                break;
            }
        }
    }
    return 0;
}
```

Output:
Duplicate elements in the array are: 2 3

4.write a c program to find max and min from an array

```
#include <stdio.h>

int main() {
    int arr[] = {10, 5, 8, 20, 3};
    int n = sizeof(arr) / sizeof(arr[0]);
    int max = arr[0];
    int min = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    printf("Maximum value in the array: %d\n", max);
    printf("Minimum value in the array: %d\n", min);

    return 0;
}
```

Output:
Maximum value in the array: 20
Minimum value in the array: 3

5.GIven a number n the task is to print the fibonacci series and sum of the series using recursion.
Input:n=10
Output=fibonacci series
0,1,1,2,3,5,8,13,21,24
sum=88

```
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
int main() {
    int n = 10;
    int sum = 0;
    printf("Fibonacci Series:\n");
    for (int i = 0; i < n; i++) {
        printf("%d, ", fibonacci(i));
        sum += fibonacci(i);
    }
    printf("\nSum of the Fibonacci Series: %d\n", sum);
    return 0;
}
```

Output:
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 search.
Example 1:arr={1,5,6,7,9,10},x=6
Output:Element found at location 2
Example1:arr={1,5,6,7,9,10},x=11
Output:Element not found in location 2

```
#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; // Element not found
```

```
}
int main() {
    int arr[] = {1, 5, 6, 7, 9, 10};
    int x = 6;
    int size = sizeof(arr) / sizeof(arr[0]);
    int result = binarySearch(arr, size, x);

    if (result != -1)
        printf("Element found at location %d\n", result);
    else
        printf("Element not found\n");
    return 0;
}
```

Output:
Element found at location 2

7.binary search in c program

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

        if (arr[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1; // Element not found
}
int main() {
    int arr[] = {2, 4, 6, 8, 10, 12, 14, 16};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 10;
    int result = binarySearch(arr, 0, n - 1, target);
    if (result == -1)
        printf("Element not found\n");
    else
        printf("Element found at index: %d\n", result);

    return 0;
```

```
}
```

Output:
Element found at index: 4

8.linear search in c programing

```
#include <stdio.h>
int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key) {
            return i;
        }
    }
    return -1;
}
int main() {
    int arr[] = {2, 5, 8, 12, 16};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 8;
    int result = linearSearch(arr, n, key);
    if (result != -1) {
        printf("Element found at index: %d", result);
    } else {
        printf("Element not found");
    }
    return 0;
}
```

Output:
Element found at index: 2

```c
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 2, 5, 6, 3};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Duplicate elements in the array are: ");
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                printf("%d ", arr[j]);
                break;
            }
        }
    }

    return 0;
}
```