1.WRITE A PROGRAM FOR LINEAR SEARCH IN C.

```
#include <stdio.h>
int linearSearch(int arr[], int size, int target) {
   for (int i = 0; i < size; ++i) {
      if (arr[i] == target) {
         return i;
      }
   }
   return -1;
}

int main() {
   int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
   int size = sizeof(arr) / sizeof(arr[0]);
   int target = 23;
   int index = linearSearch(arr, size, target);
   if (index != -1) {
      printf("Element %d found at index %d.\n", target, index);
   } else {
      printf("Element %d not found in the array.\n", target);
   }

   return 0;
}
```

SAMPLE OUTPUT:

```
Element 23 found at index 5.
```

2.WRITE A PROGRAM FOR BINARY SEARCH IN C.

```
#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;
      }
      if (arr[mid] < target) {
         left = mid + 1;
      }
      else {
         right = mid - 1;
      }
   }
   return -1;
}

int main() {
   int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
   int size = sizeof(arr) / sizeof(arr[0]);
   int target = 23;
   int index = binarySearch(arr, size, target);
   if (index != -1) {
      printf("Element %d found at index %d.\n", target, index);
   } else {
      printf("Element %d not found in the array.\n", target);
   }

   return 0;
}
```

SAMPLE OUTPUT:

```
Element 23 found at index 5.
```

3. Write a C Program to implement following operations
  a) traverse
  b) search
  c) insert
  d) delete
  e) update

a.Traverse:

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

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

    return 0;
}
```

SAMPLE OUTPUT:

```
Array elements: 1 2 3 4 5
```

b.Search:

```
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = 5;
    int element = 3;
    int position = -1;

    for (int i = 0; i < size; ++i) {
        if (arr[i] == element) {
            position = i;
            break;
        }
    }

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

    return 0;
}
```

SAMPLE OUTPUT:

```
Element 3 found at index 2.
```

C. Insert:

```
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = 5;
    int element = 10;
    int position = 2;

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

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

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

    return 0;
}
```

SAMPLE OUTPUT:

```
Before insertion:1 2 3 4 5
After insertion:1 2 10 3 4 5
```

d. Delete:

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

    printf("Before deletion:");
    for (int i = 0; i < size; ++i) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    for (int i = position; i < size - 1; ++i) {
        arr[i] = arr[i + 1];
    }
    size -= 1;

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

    return 0;
}
```

SAMPLE OUTPUT:

```
Before deletion:1 2 3 4 5
After deletion:1 2 4 5
```

e.Update:

```
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = 5;
    int element = 10;
    int position = 2;

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

    arr[position] = element;

    printf("After updating:");
```

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

    return 0;
}
```

SAMPLE OUTPUT:

```
Before updating:1 2 3 4 5
After updating:1 2 10 4 5
```

4. Writing a recursive function to calculate the factorial of a number.

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

int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);
    if (number < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        unsigned long long result = factorial(number);
        printf("The factorial of %d is %llu\n", number, result);
    }

    return 0;
}
```

SAMPLE OUTPUT:

```
Enter a number: 4
The factorial of 4 is 24
```

5. 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, 7, 8, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    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;
            }
        }
    }
    printf("\n");

    return 0;
}
```

SAMPLE OUTPUT:

```
Duplicate elements in the array are:2 3 6
```

6. Write a C Program to find Max and Min from an array elements.

```
#include <stdio.h>
int main() {
    int arr[] = {3, 5, 7, 2, 8, -1, 4, 10, 12};
    int size = sizeof(arr) / sizeof(arr[0]);
    int max = arr[0], min = arr[0];
    for (int i = 1; i < size; 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;
}
```

SAMPLE OUTPUT:

```
Maximum element: 12
Minimum element: -1
```

7. Given a number n. the task is to print the Fibonacci series and the sum of the series using recursion.

input: n=10

output: Fibonacci series

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Sum: 88

```
#include <stdio.h>
int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
void printFibonacciAndSum(int n, int *sum) {
    if (n < 0) return;

    int fib = fibonacci(n);
    printFibonacciAndSum(n - 1, sum);
    printf("%d", fib);
    if (n != 0) {
        printf(", ");
    }

    *sum += fib;
}
int main() {
    int n = 10;
    int sum = 0;

    printf("Fibonacci series:\n");
    printFibonacciAndSum(n - 1, &sum);  // Print Fibonacci series
    printf("\n");

    printf("Sum: %d\n", sum);

    return 0;
}
```

SAMPLE OUTPUT:

```
Fibonacci series:
01, 1, 2, 3, 5, 8, 13, 21, 34,
Sum: 88
```

8. 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
Example 2: arr={ 1,5,6,7,9,10},X=11
Output : Element not found at location 2

```
#include <stdio.h>
int binarySearch(int arr[], int low, int high, int x) {
   if (low <= high) {
      int mid = low + (high - low) / 2;
      if (arr[mid] == x)
         return mid;
      if (arr[mid] > x)
         return binarySearch(arr, low, mid - 1, x);
      return binarySearch(arr, mid + 1, high, x);
   }
   return -1;
}

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

   int result = binarySearch(arr, 0, size - 1, x);
   if (result != -1)
      printf("Element found at location %d\n", result);
   else
      printf("Element not found\n");

   return 0;
}
```

SAMPLE OUTPUT:

```
Element found at location 2
```