ASSIGNMENT-1

COURSE CODE: CSA0390

1.Linear search

```
#include <stdio.h>
Int linearSearch(int* arr, int size, int key)
{
    // starting traversal
    For (int I = 0; I < size; i++) {
        // checking condition
        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].

2.Binary search

```
#include <stdio.h>
Int binarySearch(int arr[], int low, int high, int x)
{
    While (low <= high) {
        Int mid = low + (high – low) / 2;
        If (arr[mid] == x)
            Return mid;
        If (arr[mid] < x)
            Low = mid + 1;
        Else
            High = mid – 1;
    }
    Return -1;
}
Int main(void)
{
    Int arr[] = { 2, 3, 4, 10, 40 };
    Int n = sizeof(arr) / sizeof(arr[0]);
    Int x = 10;
    Int result = binarySearch(arr, 0, n – 1, x);
```

```
    (result == -1) ? printf(“Element is not present”
                        “ in array”)
                : printf(“Element is present at “
                        “index %d”,
                        Result);
    Return 0;
}
```

Output

Element is present at index 3

3. Write a C Program to implement following operations

  a) traverse

  b) search

  c) insert

  d) delete

  e) update

A.)Transverse

```
#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 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 arr[] = { 12, 34, 10, 6, 40 };
    Int n = sizeof(arr) / sizeof(arr[0]);
    Int key = 40;
```

```
    Int position = findElement(arr, n, key);

    If (position == -1)
        Printf(“Element not found”);
    Else
        Printf(“Element Found at Position: %d”,
              Position + 1);

    Return 0;
}
```

Output

Element Found at Position: 5

C.) Insert

```
#include <stdio.h>
Void insertElement(int arr[], int n, int x, int pos)
{
    For (int I = n – 1; I >= pos; i--)
        Arr[I + 1] = arr[i];

    Arr[pos] = x;
}
Int main()
{
    Int arr[15] = { 2, 4, 1, 8, 5 };
    Int n = 5;
 Printf(“Before insertion : “);
    For (int I = 0; I < n; i++)
```

```
        Printf(“%d “, arr[i]);
  Printf(“\n”);
     Int x = 10, pos = 2;
    insertElement(arr, n, x, pos);
    n++;
    printf(“After insertion : “);
    for (int I = 0; I < n; i++)
        printf(“%d “, arr[i]);
     return 0;
}
```

Output

Before insertion : 2 4 1 8 5

After insertion : 2 4 10 1 8 5

D.) Delete

```
#include <stdio.h>
Int findElement(int arr[], int n, int key);
Int deleteElement(int arr[], int n, int key)
{
    // Find position of element to be deleted
    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 size, target, newValue;
    printf("Enter size: ");
    scanf("%d", &size);

    int arr[size];
    printf("Enter elements: ");
    for (int i = 0; i < size; i++) scanf("%d", &arr[i]);

    printf("Enter target: ");
    scanf("%d", &target);

    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            printf("Enter new value: ");
```

```
            scanf("%d", &newValue);
            arr[i] = newValue;
            break;
        }
    }

    printf("Updated array: ");
    for (int i = 0; i < size; i++) printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

Input:

Enter size: 5

Enter elements: 1 3 5 7 9

Enter target: 7

Enter new value: 8

Output:

Updated array: 1 3 5 8 9

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

```
#include<stdio.h>
long int multiplyNumbers(int n);
int main() {
    int n;
    printf("Enter a positive integer: ");
```

```
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, multiplyNumbers(n));
    return 0;
}

long int multiplyNumbers(int n) {
    if (n>=1)
        return n*multiplyNumbers(n-1);
    else
        return 1;
}
```

Output

```
Enter a positive integer: 6
Factorial of 6 = 720
```

5. Write a C Program to find duplicate element in an array

```
#include <stdio.h>

int findDuplicate(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                return arr[i];
            }
        }
    }
```

```
    return -1; // If no duplicate is found
}

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

    if (duplicate != -1) {
        printf("Duplicate element found: %d\n", duplicate);
    } else {
        printf("No duplicate element found\n");
    }

    return 0;
}
```

Input:

```
int arr[] = {1, 2, 3, 4, 5, 3};
```

Output:

```
Duplicate element found: 3
```

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

```
#include <stdio.h>
void findMinMax(int arr[], int size, int *min, int *max) {
    *min = *max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < *min) {
```

```
            *min = arr[i];
        }
        if (arr[i] > *max) {
            *max = arr[i];
        }
    }
}

int main() {
    int arr[] = {12, 3, 7, 1, 9, 34, 2};
    int size = sizeof(arr) / sizeof(arr[0]);
    int min, max;

    findMinMax(arr, size, &min, &max);

    printf("Minimum element: %d\n", min);
    printf("Maximum element: %d\n", max);

    return 0;
}
```

Input:

```
int arr[] = {12, 3, 7, 1, 9, 34, 2};
```

Output:

Minimum element: 1

Maximum element: 34

7. 5. 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 <= 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++) {
        int fib = fibonacci(i);
        printf("%d ", fib);
        sum += fib;
    }

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

```
    return 0;
}
```

Input:

int n = 10;

Output:

Fibonacci series:

0 1 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 size, int x) {
    int low = 0, high = size - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) return mid;
        if (arr[mid] < x) low = mid + 1;
        else high = mid - 1;
    }
```

```
    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, size, x);

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

    return 0;
}
```

Input:

For `x = 6`:

int x = 6;

Output:

Element found at location 2

For `x = 11`:

int x = 11;

Output:

Element not found