{

DATA STRUCTURE

24/7/24

1. Write a C Program to implement following operations

a.traverse

```
#include <stdio.h>
int main()
 {
int array[] = {1, 2, 3, 4, 5};
int size = sizeof(array) / sizeof(array[0]);
   printf("Elements of the array: ");
    for (int i = 0; i < size; ++i)
 {
   printf("%d ", array[i]);
    }
printf("\n");
}
```

OUTPUT

```
Elements of the array: 1 2 3 4 5
```

b.search

```
#include <stdio.h>
int main()
 {
```

```
int array[] = {10, 20, 30, 40, 50};
    int size = 5;
int key = 30;
    int found = 0;
for (int i = 0; i < size; ++i)
 {
        if (array[i] == key)
 {
found = 1;
            printf("%d  element found  %d\n", key, i);
       break;
        }
}
if (!found)
{
        printf(" %d not found", key);
}
    return 0;
}
```

OUTPUT

```
30  element found  2
```

c.insert

```
#include <stdio.h>
  int main() {
    int array[10] = {10, 20, 30, 40, 50};
    int size = 5;
    int position = 2;
      printf("Original Array: ");
    for (int i = 0; i < size; ++i) {
        printf("%d ", array[i]);
    }
    printf("\n");
    for (int i = size; i > position; --i) {
        array[i] = array[i - 1];
    }
     array[position] = element;
    size++;
     printf("Modified Array: ");
    for (int i = 0; i < size; ++i) {
        printf("%d ", array[i]);
    }
    printf("\n");
 return 0;
}
```

OUTPUT

```
Original Array: 10 20 30 40 50
Modified Array: 10 20 25 30 40 50
```

d. delete an array

```
#include <stdio.h>
 int main() {
    int array[10] = {10, 20, 30, 40, 50};
    int size = 5;

    int position = 2;

     printf("Original Array: ");

    for (int i = 0; i < size; ++i) {

        printf("%d ", array[i]);

    }

    printf("\n");

     for (int i = position; i < size - 1; ++i) {

        array[i] = array[i + 1];

    }

     size--;

     printf("Modified Array: ");

    for (int i = 0; i < size; ++i) {

        printf("%d ", array[i]);

    }

    printf("\n");

  return 0;

}
```

OUTPUT

```
Original Array: 10 20 30 40 50

Modified Array: 10 20 40 50
```

e. update

```
#include <stdio.h>

 int main() {
    int array[] = {10, 20, 30, 40, 50};

    int index = 2;

    printf("Original Array: ");

    for (int i = 0; i < 5; ++i) {

        printf("%d ", array[i]);

    }

    printf("\n");

     if (index >= 0 && index < 5) {

        array[index] = newValue;

        printf("Updated Array: ");

        for (int i = 0; i < 5; ++i) {

            printf("%d ", array[i]);

        }

        printf("\n");

    } else {

        printf("Invalid index.\n");

    }


    return 0;

}
```

OUTPUT

```
Original Array: 10 20 30 40 50

Updated Array: 10 20 35 40 50
```

2. 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 num;
     printf("Enter a non-negative integer: ");
    scanf("%d", &num);
     if (num < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        unsigned long long result = factorial(num);
        printf("Factorial of %d is %llu\n", num, result);
    }
    return 0;
}
```

OUTPUT

```
Enter a non-negative integer: 5
Factorial of 5 is 120
```

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

```
#include <stdio.h>
 int main() {
    int arr[10] = {2, 5, 6, 8, 2, 3, 7, 9, 5, 8};
    int size = 10;
    int i, j;

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

    printf("\n");

    return 0;
}
```

OUTPUT

```
Duplicate elements in the array: 2 5 8
```

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

```
#include <stdio.h>
 int main() {
    int arr[] = {7, 15, 3, 22, 10, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int max, min;
     max = 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 in the array: %d\n", max);
    printf("Minimum element in the array: %d\n", min);

    return 0;
}
```

OUTPUT

```
Maximum element in the array: 22
Minimum element in the array: 3
```

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

```
  #include <stdio.h>

void fibonacci(int n)
{
    int a = 0, b = 1, next;
    printf("Fibonacci Series: ");

    for (int i = 0; i < n; i++)
{
        printf("%d ", a);
        next = a + b;
        a = b;
        b = next;
    }
    printf("\n");
}

int main()
 {
    int n;

    printf("Enter the number of terms\n");
    if (scanf("%d", &n) != 1 || n < 1)
 {
        printf("enter a positive integer.\n");
        return 1;
    }

    fibonacci(n);

    return 0;
}
```

OUTPUT

```
Enter the number of terms

10

Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

6. You are given an array arr in increasing order. Find the element x from arr using binary search.

```
#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;
      }
            if (arr[mid] < x)
{
         left = mid + 1;
      }
```

```
             Else
 {
            right = mid - 1;
        }
    }

       return -1;
}

int main()
 {
    int arr[] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};
    int size = sizeof(arr) / sizeof(arr[0]);
    int x;

    printf("Enter the element to search: ");
    scanf("%d", &x);

    int result = binarySearch(arr, size, x);

    if (result == -1)
 {
        printf(" %d\n", x);
    } else {
        printf(" %d \n", x, result);
    }
    return 0;

}
```

OUTPUT

```
Enter the element to search: 7

Element 7 is present at index 3
```

7. linear search in c programme

```
#include <stdio.h>

 int linearSearch(int arr[], int size, int x)

 {

    for (int i = 0; i < size; i++)

 {

        if (arr[i] == x)

 {

            return i;

      }

    }

    return -1;

}

int main()

 {

    int arr[] = {12, 45, 67, 23, 56, 78, 34, 89};

    int size = sizeof(arr) / sizeof(arr[0]);
    int x;

    printf("Enter the element to search: ");
    scanf("%d", &x);

    int result = linearSearch(arr, size, x);

    if (result == -1)
 {
        printf("Element %d \n", x);
    } else {
        printf("Element %d \n", x, result);
    }

    return 0;
}
```

OUTPUT

```
Enter the element to search: 45
Enter the element to search: 45
```

8. binary search in c programme

```
#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;
        }
                if (arr[mid] < x)
 {
            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 x;
    printf("Enter the element to search: ");
    scanf("%d", &x);

    int result = binarySearch(arr, size, x);

    if (result == -1) {
        printf("Element %d is not present in the array.\n", x);
    } else {
        printf("Element %d is present at index %d.\n", x, result);
    }

    return 0;
}
```

OUTPUT

```
Enter the element to search: 8
Element 8 is present at index 2.
```