1. //Write recursive code to calculate a^n

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

int main(){
     int a, n;
     printf("Enter the base number: ");
     scanf("%d", &a);

     printf("Enter the power: ");
     scanf("%d", &n);

     printf("The power of %d raised to %d i.e. %d^%d is
     %d\n", a, n, a, n, power(a, n));
}
```

OUTPUT:

```
➜ Lab 4 git:(master) ✗ gcc src/01.c -o bin/01 -Wall
➜ Lab 4 git:(master) ✗ ./bin/01
Enter the base number: 4
Enter the power: 5
The power of 4 raised to 5 i.e. 4^5 is  1024
➜ Lab 4 git:(master) ✗
```

2. //Write recursive code to generate Fibonacci series.

```
#include<stdio.h>

int fibo(int n){
     if(n==1)
          return 0;
     else if(n == 2)
          return 1;
     else
          return fibo(n - 1) + fibo(n - 2);
}


int main(){
     int n;

     printf("Enter the number of terms: ");
     scanf("%d", &n);

     printf("The fibonacci series upto %d terms: ", n);
     for (int i = 1; i <= n; i++)
          printf("%4d", fibo(i));

     printf("\n");
}
```

OUTPUT:

```
➜  Lab 4 git:(master) ✗ gcc src/02.c -o bin/02 -Wall
➜  Lab 4 git:(master) ✗ ./bin/02
Enter the number of terms: 6
The fibonacci series upto 6 terms:    0   1   1   2   3   5
➜  Lab 4 git:(master) ✗
```

3. //Sort an array using insertion sort.

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

void InsertionSort(int *array, int n)
{
     int i, element, j;
     for (i = 1; i < n; i++) {
          element = array[i];
          j = i - 1;
          while (j >= 0 && array[j] > element)
          {
               array[j + 1] = array[j];
               j = j - 1;
          }
          array[j + 1] = element;
     }
}

int main(){

     int n;
     printf("Enter the number of elements of the array: ");
     scanf("%d", &n);

     int *arr = malloc(n * sizeof(int *));

     printf("Enter the elements of the array: ");

     for (int i = 0; i < n; i++)
          scanf("%d", (arr + i));

     printf("\n\n\nThe unsorted array: \n");
     for (int i = 0; i < n; i++)
          printf("%5d", *(arr + i));

     printf("\n");

     InsertionSort(arr, n);

     printf("\n\nThe sorted array: \n");

     for (int i = 0; i < n; i++)
          printf("%5d", *(arr + i));

     printf("\n\n");
}
```

OUTPUT:

```
➜  Lab 4 git:(master) ✗ gcc src/03.c -o bin/03 -Wall
➜  Lab 4 git:(master) ✗ ./bin/03
Enter the number of elements of the array: 6
Enter the elements of the array: 34 45 98 23 12 23


The unsorted array:
   34   45   98   23   12   23


The sorted array:
   12   23   23   34   45   98

➜  Lab 4 git:(master) ✗
```

4. //Sort an array using merge sort.

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

void merge(int arr[], int l, int m, int r)
{
     int i, j, k;
     int n1 = m - l + 1;
     int n2 = r - m;

     // Create temp arrays
     int L[n1], R[n2];

     // Copy data to temp array
     for (i = 0; i < n1; i++)
          L[i] = arr[l + i];
     for (j = 0; j < n2; j++)
          R[j] = arr[m + 1+ j];

     // Merge the temp arrays
     i = 0;
     j = 0;
     k = l;
     while (i < n1 && j < n2)
     {
          if (L[i] <= R[j])
          {
               arr[k] = L[i];
               i++;
          }
          else
          {
               arr[k] = R[j];
               j++;
          }
          k++;
     }

     // Copy the remaining elements of L[]
     while (i < n1)
     {
          arr[k] = L[i];
          i++;
          k++;
     }
     // Copy the remaining elements of R[]
     while (j < n2)
     {
          arr[k] = R[j];
```

```
            j++;
            k++;
        }
}

void MergeSort(int *arr, int l, int r)
{
        if (l < r)
        {
            int m = l+(r-l)/2;

            MergeSort(arr, l, m);
            MergeSort(arr, m+1, r);

            merge(arr, l, m, r);
        }
}

int main(){

        int n;
        printf("Enter the number of elements of the array: ");
        scanf("%d", &n);

        int *arr = malloc(n * sizeof(int *));

        printf("Enter the elements of the array: ");

        for (int i = 0; i < n; i++)
            scanf("%d", (arr + i));

        printf("\n\nThe unsorted array: \n");
        for (int i = 0; i < n; i++)
            printf("%5d", *(arr + i));

        printf("\n");

        MergeSort(arr,0,n-1);

        printf("\n\nThe sorted array: \n");
        for (int i = 0; i < n; i++)
            printf("%5d", *(arr + i));

        printf("\n\n");
}
```

OUTPUT:

```
➜  Lab 4 git:(master) ✗ gcc src/04.c -o bin/04 -Wall
➜  Lab 4 git:(master) ✗ ./bin/04
Enter the number of elements of the array: 7
Enter the elements of the array: 87 34 45 98 9 1 10


The unsorted array:
   87   34   45   98    9    1   10


The sorted array:
    1    9   10   34   45   87   98

➜  Lab 4 git:(master) ✗
```

5. //Find a number in an array using linear search.

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

int LinearSearch(int *arr, int n, int query){
     int location;
     int i = 0;
     while (i<=n && query != arr[i])
          i++;
     if (i<=n)
          location = i+1;
     else
          location = 0;
     return location;
}

int main(){

     int n,query;
     printf("Enter the number of elements of the array: ");
     scanf("%d", &n);

     int *arr = malloc(n * sizeof(int *));

     printf("Enter the elements of the array: ");

     for (int i = 0; i < n; i++)
     scanf("%d", (arr + i));

     printf("\nThe elements of array: \n");
     for (int i = 0; i < n; i++)
          printf("%5d", *(arr + i));

     printf("\n");
     printf("Enter a number to search: ");
     scanf("%d", &query);

     if (LinearSearch(arr, n - 1, query))
     {
          printf("The location of %d is index %d in array\n",
          query, LinearSearch(arr, n - 1, query));
     }
     else
          printf("%d is not inside the array.\n",query);
     printf("\n\n");
}
```

OUTPUT:

```
➜ Lab 4 git:(master) ✗ gcc src/05.c -o bin/05 -Wall
➜ Lab 4 git:(master) ✗ ./bin/05
Enter the number of elements of the array: 7
Enter the elements of the array: 87 34 45 98 9 1 10

The elements of array:
   87   34   45   98    9    1   10
Enter a number to search: 9
The location of 9 is index 5 in array


➜ Lab 4 git:(master) ✗ ./bin/05
Enter the number of elements of the array: 7
Enter the elements of the array: 87 34 45 98 9 1 10

The elements of array:
   87   34   45   98    9    1   10
Enter a number to search: 100
100 is not inside the array.
```

6. // Find a number in an array using binary search.

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

int floorValue(float num){
     if (num == (int)(num)){
          return num;
     }
     else if (num>=0){
          return (int)(num/1);
     }
     else{
          return (int)(num-1);
     }
}

int BinarySearch(int *arr, int l, int r, int query){

     int i = l;
     int j = r;
     int location;

     while (i<j)
     {
          int m = floorValue((i + j) / 2.0);
          if (query > *(arr + m))
               i = m + 1;
          else
               j = m;
     }
     if (query == *(arr + i))
          location = i + 1;
     else
          location = 0;
     return location;
}

void bubbleSort(int *arr, int n){

     for (int i = 0; i < n-1; i++)
     {
          for (int j = 0; j < n-i-1; j++)
          {
               if (*(arr + j) > *(arr + j + 1))
               {
                    int temp = *(arr + j);
                    *(arr + j) = *(arr + j + 1);
                    *(arr + j + 1) = temp;
               }
```

```
            }
        }
}

int main(){

        int n,query;
        printf("Enter the number of elements of the array: ");
        scanf("%d", &n);

        int *arr = malloc(n * sizeof(int *));

        printf("Enter the elements of the array: ");

        for (int i = 0; i < n; i++)
            scanf("%d", (arr + i));

        printf("\nThe elements of array: \n");
        for (int i = 0; i < n; i++)
            printf("%5d", *(arr + i));

        printf("\n");

        bubbleSort(arr, n);

        printf("\n\nThe sorted array: \n");
        for (int i = 0; i < n; i++)
            printf("%5d", *(arr + i));

        printf("\n\n");

        printf("Enter a number to search: ");
        scanf("%d", &query);

        if (BinarySearch(arr, 0, n-1, query))
        {
            printf("The location of %d is index %d in array\n",
            query, BinarySearch(arr, 0, n-1, query));
        }
        else
            printf("%d is not inside the array.\n",query);

        printf("\n\n");
}
```

OUTPUT:

```
➜  Lab 4 git:(master) ✗ gcc src/06.c -o bin/06 -Wall
➜  Lab 4 git:(master) ✗ ./bin/06
Enter the number of elements of the array: 7
Enter the elements of the array: 45 34 56 23 12 87 09

The elements of array:
   45   34   56   23   12   87    9


The sorted array:
    9   12   23   34   45   56   87

Enter a number to search: 45
The location of 45 is index 5 in array


➜  Lab 4 git:(master) ✗ ./bin/06
Enter the number of elements of the array: 7
Enter the elements of the array: 45 34 56 23 12 87 09

The elements of array:
   45   34   56   23   12   87    9


The sorted array:
    9   12   23   34   45   56   87

Enter a number to search: 98
98 is not inside the array.


➜  Lab 4 git:(master) ✗
```