1. Linear Search in an Array.

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

int LinearSearch(int*, int, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }
    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    int key,key_pos;
    printf("Enter the Element you want to Linear Search in the Array : ");
    scanf(" %d",&key);

    key_pos = LinearSearch(arr , length , key);

    if( key_pos == -1){
        printf("The Key Value ( %d ) is NOT Present in the Array. \n",key);
    }
    else{
        printf("The Key Value ( %d ) is Present at the index ( %d ) of the Array. \n",key,key_pos);
    }

    return(0);
}

int LinearSearch(int A[], int Len, int K){
    for(int i=0; i<Len; i++){
        if(A[i]==K){
            return(i);
        }
    }
    return(-1);
}
```

```
void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 6
Input Value at index [1] : 7
Input Value at index [2] : -3
Input Value at index [3] : 4
Input Value at index [4] : 1

The Array you created is :
[ 6, 7, -3, 4, 1 ]
Enter the Element you want to Linear Search in the Array : 4
The Key Value ( 4 ) is Present at the index ( 3 ) of the Array.
```

2. Binary Search in an Array. (Recursive)

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

int RecBinarySearch(int*, int, int, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }
    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    for (int i=0 ; i<length-1;i++){
        if(arr[i]>arr[i+1]){
            printf("The Input Array is not sorted in ascending order.\nCannot perform Binary Search.");
            return(0);
        }
    }
    int key,key_pos;
    printf("Enter the Element you want to Rec. Binary Search in the Array : ");
    scanf(" %d",&key);

    key_pos = RecBinarySearch(arr, 0, length-1, key);

    if( key_pos == -1){
        printf("The Key Value ( %d ) is NOT Present in the Array. \n",key);
    }
    else{
        printf("The Key Value ( %d ) is Present at the index ( %d ) of the Array. \n",key,key_pos);
    }


    return(0);
}
```

```
int RecBinarySearch(int A[], int lower, int upper, int K){
    int middle = (lower+upper)/2;
    if(lower==upper){
        if(A[middle] == K){
            return(middle);
        }
        else{
            return(-1);
        }
    }
    else{
        if(A[middle]==K){
            return(middle);
        }
        else{
            if(A[middle]>K){
                return(RecBinarySearch(A,lower,middle-1,K));
            }
            else{
                return(RecBinarySearch(A,middle+1,upper,K));
            }
        }
    }

}

void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output (1) :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : -11
Input Value at index [1] : 0
Input Value at index [2] : 4
Input Value at index [3] : 7
Input Value at index [4] : 47

The Array you created is :
[ -11, 0, 4, 7, 47 ]
Enter the Element you want to Rec. Binary Search in the Array : 4
The Key Value ( 4 ) is Present at the index ( 2 ) of the Array.
```

Output (2) :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 74
Input Value at index [1] : 47
Input Value at index [2] : 10
Input Value at index [3] : 7979
Input Value at index [4] : -52

The Array you created is :
[ 74, 47, 10, 7979, -52 ]
The Input Array is not sorted in ascending order.
Cannot perform Binary Search.
```

3. Binary Search in an Array. (Non Recursive)

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

int NonRecBinarySearch(int*, int, int, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }

    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    for (int i=0 ; i<length-1;i++){
        if(arr[i]>arr[i+1]){
            printf("The Inout Array is not sorted in ascending order.\nCannot perform Binary Search.");
            return(0);
        }
    }
    int key,key_pos;
    printf("Enter the Element you want to Non Rec. Binary Search in the Array : ");
    scanf(" %d",&key);

    key_pos = NonRecBinarySearch(arr, 0, length-1, key);

    if( key_pos == -1){
        printf("The Key Value ( %d ) is NOT Present in the Array. \n",key);
    }
    else{
        printf("The Key Value ( %d ) is Present at the index ( %d ) of the Array. \n",key,key_pos);
    }

    return(0);
}

int NonRecBinarySearch(int A[], int lower, int upper, int K){
    int looplen = upper+lower;
    for(int i=0;i<=log(looplen)+1;i++){
        int middle = (lower+upper)/2;
        if(K==A[middle]){
            return(middle);
        }
        else if(K < A[middle]){
            upper=middle-1;
        }
        else{
            lower=middle+1;
        }
    }
    return(-1);
}

void PrintArray(int arr[],int length){
    printf("\nThe Array you created is :\n");
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output (1) :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 11
Input Value at index [1] : -70
Input Value at index [2] : 420
Input Value at index [3] : 143
Input Value at index [4] : 69

The Array you created is :

The Array you created is :
[ 11, -70, 420, 143, 69 ]
The Inout Array is not sorted in ascending order.
Cannot perform Binary Search.
```

Output (2) :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : -47
Input Value at index [1] : -11
Input Value at index [2] : 0
Input Value at index [3] : 4
Input Value at index [4] : 7

The Array you created is :

The Array you created is :
[ -47, -11, 0, 4, 7 ]
Enter the Element you want to Non Rec. Binary Search in the Array : 7
The Key Value ( 7 ) is Present at the index ( 4 ) of the Array.
```

4. Merge Sort in an Array.

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

int Merging(int*, int, int, int, int);
int MergeSort(int*, int, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }
    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);
    printf("\nApplying The Merge Sort Algorithm ...\n");
    MergeSort(arr,0,length);
    printf("\nThe Sorted Array is :\n");
    PrintArray(arr,length);
    return(0);
}

int Merging(int A[], int low1, int up1, int low2, int up2){
    int art1[up1-low1+1],art2[up2-low2+1],length=up2-low1+1;
    for(int i=0; i<up1-low1+1; i++){
        art1[i] = A[low1+i];
        art2[i] = A[low2+i];
    }
    int i=0,j=0,k=0;
    while(i<up1-low1+1 && j<up2-low2+1){
        if(art1[i] < art2[j]){
            A[k+low1] = art1[i++];
        }
        else{
            A[k+low1] = art2[j++];
        }
        k++;
    }
    while(i<up1-low1+1){
        A[k+low1] = art1[i++];
        k++;
    }
    while(j<up2-low2+1){
        A[k+low1] = art2[j++];
        k++;
    }
}

int MergeSort(int A[], int low, int up){
    int mid = ( low + up ) / 2 ;
    if ( low == up ){
        return(A[low]);
    }
    else{
        MergeSort(A,low,mid);
        MergeSort(A,mid+1,up);
        Merging(A,low,mid,mid+1,up);
    }
}

void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 6
Input Value at index [1] : 7
Input Value at index [2] : -3
Input Value at index [3] : 4
Input Value at index [4] : 1

The Array you created is :
[ 6, 7, -3, 4, 1 ]

Applying The Merge Sort Algorithm ...

The Sorted Array is :
[ -3, 1, 4, 6, 7 ]
```

5. Quick Sort in an Array.

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

void swap(int*, int*);
int Partition(int*, int, int);
int QuickSort(int*, int, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }

    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    printf("\nApplying The Quick Sort Algorithm ...\n");
    printf("The Sorted Array is : \n");
    QuickSort(arr,0,length-1);

    PrintArray(arr,length);

    return(0);
}

void swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

int Partition(int A[], int lo, int hi){
    int pivot,i,j;
    pivot = A[lo];
    i = lo;
    for ( j = i+1 ; j <= hi ; j++ ){
        if( pivot >= A[j] ){
            i++;
            swap(&A[i],&A[j]);
        }
    }
    swap(&A[lo],&A[i]);
    return(i);
}

int QuickSort(int A[], int low, int up){
    if(low<up){
        int p = Partition(A, low, up);
        QuickSort(A,low,p-1);
        QuickSort(A,p+1,up);
    }

}

void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 36
Input Value at index [1] : -45
Input Value at index [2] : 10
Input Value at index [3] : 25
Input Value at index [4] : 77

The Array you created is :
[ 36, -45, 10, 25, 77 ]

Applying The Quick Sort Algorithm ...
The Sorted Array is :
[ -45, 10, 25, 36, 77 ]
```

6. Insertion Sort in an Array.

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

int swap(int*, int*);
int InsertionSort(int*, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }

    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    printf("\nApplying The Insertion Sort Algorithm ...\n");
    printf("The Sorted Array is : \n");
    InsertionSort(arr,length);
    PrintArray(arr,length);

    return(0);
}

int swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

int InsertionSort(int A[],int len){
    if(len>1){
        InsertionSort(A,len-1);
        int p,q,temp;
        q=len-1;
        p=q-1;
        while(p!=-1){
            if(A[p]>A[q]){
                swap(&A[p],&A[q]);
                p--;
                q--;
            }
            else{
                return(0);
            }
        }
    }
    return(0);
}

void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 9
Input Value at index [1] : 8
Input Value at index [2] : 7
Input Value at index [3] : 1
Input Value at index [4] : 2

The Array you created is :
[ 9, 8, 7, 1, 2 ]

Applying The Insertion Sort Algorithm ...
The Sorted Array is :
[ 1, 2, 7, 8, 9 ]
```

7. Selection Sort in an Array.

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

int swap(int*, int*);
int SelectionSort(int*, int, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }

    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    printf("\nApplying The Selection Sort Algorithm ...\n");
    printf("The Sorted Array is : \n");
    SelectionSort(arr,0,length);
    PrintArray(arr,length);

    return(0);
}

int swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

```
int SelectionSort(int A[],int low, int hi){
   int minimum = A[low],index = low;
   if(low == hi){
      return(0);
   }
   for(int i=low+1;i<hi;i++){
      if(A[i] < minimum){
         minimum = A[i];
         index = i;
      }
   }
   swap(&A[low],&A[index]);
   SelectionSort(A,low+1,hi);
}

void PrintArray(int arr[],int length){
   if(length<=0){
      printf("NULL");
      exit(0);
   }
   else{
      printf("[");
   }
   for(int i=0;i<length;i++){
      printf(" %d",arr[i]);
      if(i!=length-1){
         printf(",");
      }
      else if(i==length-1){
         printf(" ]\n");
      }
   }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 12
Input Value at index [1] : 20
Input Value at index [2] : 50
Input Value at index [3] : 33
Input Value at index [4] : -5

The Array you created is :
[ 12, 20, 50, 33, -5 ]

Applying The Selection Sort Algorithm ...
The Sorted Array is :
[ -5, 12, 20, 33, 50 ]
```

8. Bubble Sort in an Array. (with Flag)

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

int swap(int*,int*);
void FlagBubbleSort(int*, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }

    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    printf("\nApplying The Flag Bubble Sort Algorithm ...\n");
    printf("The Sorted Array is : \n");
    FlagBubbleSort(arr,length);
    PrintArray(arr,length);

    return(0);
}

int swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

void FlagBubbleSort(int A[],int len){
    int flag=1;
    for(int i=0;i<len;i++){
        flag=1;
        for(int j=0;j<len-i-1;j++){
            if(A[j]>=A[j+1]){
                swap(&A[j],&A[j+1]);
                flag=0;
            }

        }
        if(flag){
            break;
        }
    }
}

void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 1
Input Value at index [1] : 5
Input Value at index [2] : 15
Input Value at index [3] : 11
Input Value at index [4] : 45

The Array you created is :
[ 1, 5, 15, 11, 45 ]

Applying The Flag Bubble Sort Algorithm ...
The Sorted Array is :
[ 1, 5, 11, 15, 45 ]
```

9. Bubble Sort in an Array. (without Flag)

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

int swap(int*,int*);
void nonFlagBubbleSort(int*, int);
void PrintArray(int*, int);

int main(){
    int length;
    printf("Enter the Size of the Array you want to create : ");
    scanf(" %d",&length);
    int arr[length];

    printf("Enter the numerical values into the Array\n");
    for(int i=0;i<length;i++){
        printf("Input Value at index [%d] : ",i);
        scanf(" %d",&arr[i]);
    }

    printf("\nThe Array you created is :\n");
    PrintArray(arr,length);

    printf("\nApplying The Non Flag Bubble Sort Algorithm ...\n");
    printf("The Sorted Array is : \n");
    nonFlagBubbleSort(arr,length);
    PrintArray(arr,length);

    return(0);
}

int swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

void nonFlagBubbleSort(int A[],int len){
    for(int i=0;i<len;i++){
        for(int j=0;j<len-i-1;j++){
            if(A[j]>=A[j+1]){
                swap(&A[j],&A[j+1]);
            }
        }
    }

}
```

```
void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter the Size of the Array you want to create : 5
Enter the numerical values into the Array
Input Value at index [0] : 90
Input Value at index [1] : 40
Input Value at index [2] : 10
Input Value at index [3] : 50
Input Value at index [4] : 55

The Array you created is :
[ 90, 40, 10, 50, 55 ]

Applying The Non Flag Bubble Sort Algorithm ...
The Sorted Array is :
[ 10, 40, 50, 55, 90 ]
```

10.Heap Sort in an Array.

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

void swap(int*, int*);
void max_heapify(int*, int, int);
void build_heap(int*, int);
void heap_sort(int*, int);
void PrintArray(int*, int);

int main(){
    int n;
    printf("Enter length of the Array : ");
    scanf(" %d", &n);

    int arr[n];
    printf("\nEnter elements in the Array\n");
    for (int i = 0; i < n; i++){
            printf("Input Value at index [%d] : ",i);
            scanf(" %d",&arr[i]);
    }

    printf("\nThe Input Array is\n");
    PrintArray(arr,n);

    heap_sort(arr, n);

    printf("\nThe Array after Heap sorting\n");
    PrintArray(arr,n);
}

void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

void max_heapify(int arr[], int n, int i){
    int left = (2 * i) + 1;
    int right = (2 * i) + 2;
    int largest = i;
    if(left < n && arr[left] > arr[i]){
        largest = left;
    }
    if (right < n && arr[right] > arr[largest]){
        largest = right;
    }
    if (largest != i){
        swap(&arr[i], &arr[largest]);
        max_heapify(arr, n, largest);
    }
}

void build_heap(int arr[], int n){
    for(int i = (n / 2) ; i >= 0; i--){
        max_heapify(arr, n, i);
    }
}

void heap_sort(int arr[], int n){
    build_heap(arr, n);
    for(int i = n-1 ; i >= 0; i--){
        swap(&arr[0], &arr[i]);
        max_heapify(arr, i, 0);
    }
}

void PrintArray(int arr[],int length){
    if(length<=0){
        printf("NULL");
        exit(0);
    }
    else{
        printf("[");
    }
    for(int i=0;i<length;i++){
        printf(" %d",arr[i]);
        if(i!=length-1){
            printf(",");
        }
        else if(i==length-1){
            printf(" ]\n");
        }
    }
}
```

Output :

```
Enter length of the Array : 5

Enter elements in the Array
Input Value at index [0] : 27
Input Value at index [1] : 6
Input Value at index [2] : 11
Input Value at index [3] : 47
Input Value at index [4] : 0

The Input Array is
[ 27, 6, 11, 47, 0 ]

The Array after Heap sorting
[ 0, 6, 11, 27, 47 ]
```

11. Queue Data Structure with all operations.

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

int IsFull(int*, int*, int*, int);
int IsEmpty(int*, int*, int*, int);
void Peek(int*, int*, int*, int);
void Enqueue(int*, int*, int*, int);
void Dequeue(int*, int*, int*, int);

int main(){
    int length,elements,operation;

    printf("Enter the size of the Queue you want to create : ");
    scanf(" %d",&length);

    int queue[length];

    printf("Enter the number of values you want to input in queue : ");
    scanf(" %d",&elements);

    if(elements >= 0 && elements <= length){
        printf("Input the Values\n");
        for(int i=0;i<elements;i++){
            printf("Input at index ( %d ) : ",i);
            scanf(" %d",&queue[i]);
        }
    }
    else{
        printf("Cannot Insert ...");
        return(0);
    }

    int front, rear;
    if(elements == 0){
        front = rear = -1;
    }
    else{
        front = 0;
        rear = elements-1;
    }

    while(1){
        printf("\n---------------");
        printf("\nChoose the operations to perform on Queue Data Structure.\n1. IsFull\n2. IsEmpty\n3. Peek\n4. Enqueue\n5. Dequeue\n\n0. To Exit the Program.\nEnter Here : ");
        scanf(" %d",&operation);

        int boolVal;
        switch (operation)
```

```
{
case 1:
    boolVal = IsFull(queue,&front,&rear,length);
    if(boolVal == 1){
        printf("\nQueue is Full.");
    }
    else{
        printf("\nQueue is NOT Full.");
    }
    break;
case 2:
    boolVal = IsEmpty(queue,&front,&rear,length);
    if(boolVal == 1){
        printf("\nQueue is Empty.");
    }
    else{
        printf("\nQueue is NOT Empty.");
    }
    break;
case 3:
    if(IsEmpty(queue,&front,&rear,length)){
        printf("\nPeek cannot performed on Empty Queue.");
    }
    else{
        Peek(queue,&front,&rear,length);
    }
    break;
case 4:
    if(IsFull(queue,&front,&rear,length)){
        printf("Enqueue cannot be performed. The Queue is already Full.");
    }
    else{
        Enqueue(queue,&front,&rear,length);
    }
    break;
case 5:
    if(IsEmpty(queue,&front,&rear,length)){
        printf("Dequeue cannot be performed. The Queue is Empty.");
    }
    else{
        Dequeue(queue,&front,&rear,length);
    }
    break;
case 0:
    printf("\nExiting Program . . .\n");
    return(0);
    break;

default:
    printf("Uh Oh! Invalid Input.\nExiting Program . . .");
    return(0);
}
```

```
    }

    return(0);
}

int IsFull(int Q[], int *front, int *rear, int size){
    if(*rear == size-1){
        return(1);
    }
    return(0);
}

int IsEmpty(int Q[], int *front, int *rear, int size){
    if(*rear==-1 || *rear < *front){
        return(1);
    }
    return(0);
}

void Peek(int Q[], int *front, int *rear, int size){
    printf("\nThe Current Front Element is %d.",Q[*front]);
}

void Enqueue(int Q[], int *front, int *rear, int size){
    *rear += 1;
    int value;
    printf("\nInput at index ( %d ) : ",*rear);
    scanf(" %d",&Q[*rear]);
}

void Dequeue(int Q[], int *front, int *rear, int size){
    printf("\nThe Deleted Element is %d.",Q[*front]);
    *front += 1;
}
```

Output :

```
Enter the size of the Queue you want to create : 5
Enter the number of values you want to input in queue : 2
Input the Values
Input at index ( 0 ) : 10
Input at index ( 1 ) : 20

---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 4
```

```
Input at index ( 2 ) : 25

---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 5

The Deleted Element is 10.
---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 3

The Current Front Element is 20.
---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 4

Input at index ( 3 ) : 8

---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 4
```

```
Input at index ( 4 ) : 9

---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 4
Enqueue cannot be performed. The Queue is already Full.
---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 1

Queue is Full.
---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 3

The Current Front Element is 20.
---------------
Choose the operations to perform on Queue Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Enqueue
5. Dequeue

0. To Exit the Program.
Enter Here : 0

Exiting Program . . .
```

12.Stack Data Structure with all operations.

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

int IsFull(int*, int*, int);
int IsEmpty(int*, int*, int);
int Peek(int*, int*, int);
void Push(int*, int*, int);
int Pop(int*, int*, int);
int StackDisplay(int*, int*, int);

int main(){
    int length,operation;

    printf("Enter the size of the Stack you want to create : ");
    scanf(" %d",&length);

    int stack[length];

    int top=-1;

    while(1){
        printf("\n---------------");
        printf("\nChoose the operations to perform on Stack Data Structure.\n1. IsFull\n2. IsEmpty\n3. Peek\n4. Push\n5. Pop\n6. Display\n\n0. To Exit the Program.\nEnter Here : ");
        scanf(" %d",&operation);

        int boolVal,value,pop_value;
        switch (operation)
        {
        case 1:
            boolVal = IsFull(stack,&top,length);
            if(boolVal){
                printf("\nStack is Full.");
            }
            else{
                printf("\nStack is NOT Full.");
            }
            break;
        case 2:
            boolVal = IsEmpty(stack,&top, length);
            if(boolVal){
                printf("Stack is Empty.");
            }
            else{
                printf("Stack is NOT Empty.");
            }
            break;
        case 3:
            Peek(stack,&top,length);
            break;
        case 4:
            Push(stack,&top,length);
            break;
        case 5:
            if(!IsEmpty(stack,&top,length)){
                pop_value = Pop(stack,&top,length);
                printf("\nThe Popped element is %d",pop_value);
            }
            else{
                Pop(stack,&top,length);
            }
            break;
        case 6:
            StackDisplay(stack,&top,length);
            break;
        case 0:
            printf("\nExiting Program . . .\n");
            return(0);
            break;

        default:
            printf("Uh Oh! Invalid Input.\nExiting Program . . .");
            return(0);
        }
    }

    return(0);
}

int IsFull(int S[], int *top, int size){
    if(*top == size - 1){
        return(1);
    }
    return(0);
}

int IsEmpty(int S[], int *top, int size){
    if(*top == -1){
        return(1);
    }
    return(0);
}

int Peek(int S[], int *top, int size){
    if(IsEmpty(S,&*top,size)){
        printf("\nThe Stack is Empty. There is no element.");
        return(0);
    }
    printf("\nThe top element is %d",S[*top]);
}
```

```
void Push(int S[], int *top, int size){
    int value;
    if(IsFull(S, &*top, size)){
        printf("\nStack Overflow.");
    }
    else{
        printf("\nEnter the value to Push : ");
        scanf(" %d",&value);
        *top+=1;
        S[*top]=value;
    }
}

int Pop(int S[], int *top, int size){
    if(IsEmpty(S, &*top, size)){
        printf("\nStack Underflow.");
    }
    else{
        int pop_value;
        pop_value=S[*top];
        *top-=1;
        return(pop_value);
    }
}

int StackDisplay(int S[], int *top, int size){
    if(IsEmpty(S,&*top,size)){
        printf("\nStack is Empty.");
        return(0);
    }
    else{
        for(int i=size-1;i>=0;i--){
            if(i<=*top){
                printf("\n----\n%d",S[i]);
            }
            else{
                printf("\n----\n");
            }
        }
        printf("\n----\n");
    }
}
```

Output :

```
Enter the size of the Stack you want to create : 2

---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 4

Enter the value to Push : 10

---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 4

Enter the value to Push : 20

---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 4

Stack Overflow.
---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
```

```
6. Display

0. To Exit the Program.
Enter Here : 6

----
20
----
10
----

---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 1

Stack is Full.
---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 5

The Popped element is 20
---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 6
```

```
----

----
10
----

---------------
Choose the operations to perform on Stack Data Structure.
1. IsFull
2. IsEmpty
3. Peek
4. Push
5. Pop
6. Display

0. To Exit the Program.
Enter Here : 0

Exiting Program . . .
```

13.Linked List Data Structure with all operations.

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

struct node{
    int data;
    struct node *next;
};
typedef struct node node;
node *start = NULL;

node *getnode(int);
void createLL(int);
int countnode();
void insertionLL();
void deletionLL();
void displayLL();
void searchingLL();

int main(){
    int length,operation,value;

    printf("Enter the length of Linked List you want to create : ");
    scanf(" %d",&length);

    createLL(length);

    while(1){
        printf("\n---------------");
        printf("\nChoose the operations to perform on Linked List Data Structure.\n1. Count Node\n2. Insertion\n3. Deletion\n4. Searching\n5. Display\n\n0. To Exit the Program.\nEnter Here : ");
        scanf(" %d",&operation);

        int returnval;
        switch(operation){
        case 1:
            returnval = countnode();
            printf("\nTotal number of nodes in Linked List is %d",returnval);
            break;
        case 2:
            insertionLL();
            break;
        case 3:
            if(countnode() == 0){
                printf("\nLinked List Underflow.");
            }
            else{
                deletionLL();
            }
            break;
        case 4:
            if(countnode() == 0){
                printf("The Linked List is Empty, Cannot Search!");
            }
            else{
                printf("\nEnter the data to search in the Linked List : ");
                scanf(" %d",&value);
                searchingLL(value);
            }
            break;
        case 5:
            displayLL();
            break;
        case 0:
            printf("\nExiting Program . . .\n");
            return(0);
            break;
        default:
            printf("Uh Oh! Invalid Input.\nExiting Program . . .");
            return(0);
        }

    }

    return(0);
}

node *getnode(int ctr){
    node *newnode;
    newnode = (node*)malloc(sizeof(node));
    printf("Enter the data in node %d : ",ctr);
    scanf(" %d",&newnode->data);
    newnode->next = NULL;
    return(newnode);
}

void createLL(int n){
    node *temp,*last;
    int ctr=0;
    while(ctr<n){
        temp = getnode(++ctr);
        if(start == NULL){
            start = temp;
        }
        else{
            last->next=temp;
        }
        last=temp;
    }

}
```

```
int countnode(){
    node *temp;
    temp=start;

    int count=0;
    while(temp!=NULL){
        count++;
        temp=temp->next;
    }
    return(count);
}

void insertionLL(){
    int pos,value,nodectr,i=1;
    node *prev,*temp;
    prev = temp = start;
    nodectr = countnode();
    printf("\nEnter position of Insertion.\n1 is beginning and %d is last position.\nEnter here : ",nodectr+1);
    scanf(" %d",&pos);
    if(pos > 0 && pos <= nodectr+1){
        node *insertnode = getnode(pos);
        while(i < pos){
            prev = temp;
            temp = temp->next;
            ++i;
        }
        if(pos == 1){
            insertnode->next = start;
            start = insertnode;
        }
        else{
            insertnode->next = prev->next;
            prev->next = insertnode;
        }
    }
    else{
        printf("\nInvalid Position of Insertion.");
    }

}

void deletionLL(){
    int delpos,nodectr,i=1;
    node *prev,*temp;
    prev = temp = start;
    nodectr = countnode();
    printf("\nEnter the position of Deletion.\n1 is beginning and %d is last position.\nEnter here : ",nodectr);
    scanf(" %d",&delpos);
    if(delpos > 0 && delpos <= nodectr){
        while(i<delpos){
            prev = temp;
            temp = temp->next;
            ++i;
        }
        if(delpos==1){
            start = temp->next;
        }
        else{
            prev->next = temp->next;
        }
        free(temp);

    }
    else{
        printf("\nInvalid Position of Deletion.");
    }
}

void displayLL(){
    node *temp;
    temp = start;
    if(start == NULL){
        printf("\n( START = %x ) -> NULL",start);
    }
    else{
        printf("\n( START = %x )",start);
        while(temp != NULL){
            printf(" -> [ %d | %x ]",temp->data,temp->next);
            temp = temp->next;
        }
    }
}

void searchingLL(int value){
    node *temp;
    int ctr=0,flag=1;
    temp = start;

    while(temp != NULL){
        ++ctr;
        if( temp->data == value){
            printf("\n%d is present at node %d",value,ctr);
            flag=0;
        }
        temp = temp->next;
    }
    if(flag){
        printf("\n%d is not present in the Linked List.",value);
    }
}
```

Output :

```
Enter the length of Linked List you want to create : 4
Enter the data in node 1 : 123
Enter the data in node 2 : 25
Enter the data in node 3 : -44
Enter the data in node 4 : 47

---------------
Choose the operations to perform on Linked List Data Structure.
1. Count Node
2. Insertion
3. Deletion
4. Searching
5. Display

0. To Exit the Program.
Enter Here : 4

Enter the data to search in the Linked List : 47

47 is present at node 4
---------------
Choose the operations to perform on Linked List Data Structure.
1. Count Node
2. Insertion
3. Deletion
4. Searching
5. Display

0. To Exit the Program.
Enter Here : 3

Enter the position of Deletion.
1 is beginning and 4 is last position.
Enter here : 2

---------------
Choose the operations to perform on Linked List Data Structure.
1. Count Node
2. Insertion
3. Deletion
4. Searching
5. Display

0. To Exit the Program.
Enter Here : 5

( START = cc29b8 ) -> [ 123 | cc29d8 ] -> [ -44 | cc29e8 ] -> [ 47 | 0 ]
---------------
Choose the operations to perform on Linked List Data Structure.
1. Count Node
2. Insertion
```

```
3. Deletion
4. Searching
5. Display

0. To Exit the Program.
Enter Here : 2

Enter position of Insertion.
1 is beginning and 4 is last position.
Enter here : 4
Enter the data in node 4 : 777

---------------
Choose the operations to perform on Linked List Data Structure.
1. Count Node
2. Insertion
3. Deletion
4. Searching
5. Display

0. To Exit the Program.
Enter Here : 1

Total number of nodes in Linked List is 4
---------------
Choose the operations to perform on Linked List Data Structure.
1. Count Node
2. Insertion
3. Deletion
4. Searching
5. Display

0. To Exit the Program.
Enter Here : 0

Exiting Program . . .
```