```
#include<iostream>
using namespace std;
void bubbleSort();
void insertionSort();
void selectionSort();
void mergeSort();
void quickSort();
void display(int,int []);
int linearSearch();
int binarySearch();
void MergeSort(int [],int );
void Merge(int [],int [],int, int [],int);
void sort(int [],int ,int );
int partition(int [],int ,int);
int BinarySearch(int[], int, int, int );
int main() {
   int choice;
   bool check=true;
```

```
cout<<"1. Bubble Sort "<<endl;
cout<<"2. Insertion Sort "<<endl;
cout<<"3. Selection Sort "<<endl;
cout<<"4. Merge Sort "<<endl;
cout<<"5. Quick Sort "<<endl;
cout<<"6. Linear Sort "<<endl;
cout<<"7. Binary Sort "<<endl;
cout<<"8. Exit "<<endl;
while(check){
    cout<<"Enter the choice: ";
    cin>>choice;
    switch (choice){
        case 1:
            bubbleSort();
            break;
        case 2:
            insertionSort();
            break;
```

```
            case 3:
                selectionSort();
                break;
            case 4:
                mergeSort();
                break;
            case 5:
                quickSort();
                break;
            case 6:
                if(linearSearch()!=-1)
                    cout<<"Number is present at:
"<<linearSearch()<<endl;
                else
                            cout<<"Number is not
present "<<endl;
                break;
            case 7:
```

```
            if(binarySearch()!=-1)
                cout<<"Number is present at:
"<<binarySearch()<<endl;
            else
                        cout<<"Number is not
present "<<endl;
            break;
        case 8:
            check= false;
            break;
        default:
            cout<<"Enter correct choice: ";
      }
   }
   return 0;
}
void bubbleSort(){
   int temp,count=1;
```

```
int size;
cout<<"Enter the size: ";
cin>>size;
int list[size];
cout<<"Enter list to be sorted: "<<endl;
for(int i=0;i<size;i++)
    cin>>list[i];
cout<<"Before Sorting: "<<endl;
for(int i=0;i<size;i++){
 cout<<list[i]<<" ";
}while(count!=size){
count=1;
for(int i=1;i<size;i++){
    for(int j=i-1;j<i;j++){
        if(list[j]>list[i]){
            temp=list[j];
            list[j]=list[i];
            list[i]=temp;
```

```
            } else
                    count++;
        }
    }
}
  cout<<"\nBy Bubble Sort: "<<endl;
   display(size,list);
}
void display(int size,int arr[]){
   for(int i=0;i<size;i++){
      cout<<arr[i]<<" ";
   }
   cout<<endl;
}
void insertionSort(){
   int size;
   cout<<"Enter the size: ";
   cin>>size;
```

```
  int list[size];
  cout<<"Enter list to be sorted: "<<endl;
  for(int i=0;i<size;i++)
     cin>>list[i];
    int temp;
for(int i=1;i<size;i++){
for(int j=0;j<i;j++){
 if(list[j]>list[i]){
    temp=list[i];
   for(int k=i;k>j;k--){
     list[k]=list[k-1];
     }
          list[j]=temp;
    }
  }
}
  cout<<"\nBy Insertion Sort: "<<endl;
  display(size,list);
```

```
}
void selectionSort(){
   int size,min,temp,pos=0,count=0;
   cout<<"Enter the size: ";
   cin>>size;
   int list[size];
   cout<<"Enter list to be sorted: "<<endl;
   for(int i=0;i<size;i++)
      cin>>list[i];
   for(int i=0;i<size;i++){
      min=list[i];
   for(int k=i;k<size-1;k++){
         if(min>list[k+1]){
            min=list[k+1];
            pos=k+1;
            count=1;
         }
      }if(count!=0){
```

```
            count=0;
        temp=list[i];
        list[i]=min;
        list[pos]=temp;
        cout<<"After "<<i<<"loop"<<endl;
        display(size,list);}
    }
    display(size,list);
}
void mergeSort(){
    int size;
    cout<<"Enter the size: ";
    cin>>size;
    int list[size];
    cout<<"Enter list to be sorted: "<<endl;
    for(int i=0;i<size;i++)
        cin>>list[i];
    MergeSort(list,size);
```

```
}
void MergeSort(int *A,int n) {
   int mid,i, *L, *R;
   if(n < 2) return; // initial condition.
   mid = n/2;  // find the mid index.
   // create left and right subarrays
   // mid elements (from index 0 till
mid-1) should be part of left sub-array
   // and (n-mid) elements (from mid to
n-1) will be part of right sub-array
   L = (int*)malloc(mid*sizeof(int));
   R = (int*)malloc((n- mid)*sizeof(int));
   for(i = 0;i<mid;i++) L[i] = A[i]; //
creating left subarray
   for(i = mid;i<n;i++) R[i-mid] = A[i]; //
creating right subarray
   MergeSort(L,mid);  // sorting the left
subarray
```

```
   MergeSort(R,n-mid);  // sorting the
right subarray
   Merge(A,L,mid,R,n-mid);  // Merging L
and R into A as sorted list.
   free(L);
   free(R);
}
void Merge(int *A,int *L,int
leftCount,int *R,int rightCount) {
   int i,j,k;
   // i - to mark the index of left
aubarray (L)
   // j - to mark the index of right
sub-raay (R)
   // k - to mark the index of merged
subarray (A)
   i = 0; j = 0; k =0;
   while(i<leftCount && j< rightCount) {
```

```
        if(L[i] < R[j]) A[k++] = L[i++];
        else  A[k++] = R[j++];
    }
    while(i < leftCount) A[k++] = L[i++];
    while(j < rightCount) A[k++] = R[j++];
    display(leftCount+rightCount,A);
}
void quickSort(){
    int size;
    cout<<"Enter the size: ";
    cin>>size;
    int list[size];
    cout<<"Enter list to be sorted: "<<endl;
    for(int i=0;i<size;i++)
        cin>>list[i];
    sort(list,0,size-1);
}
int partition(int arr[], int low, int high){
```

```
  int pivot = arr[high];
  int i = (low-1); // index of smaller
element
  for (int j=low; j<high; j++){
     // If current element is smaller than
or
     // equal to pivot
     if (arr[j] <= pivot){
        i++;
        // swap arr[i] and arr[j]
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
     }
  }
  // swap arr[i+1] and arr[high] (or pivot)
  int temp = arr[i+1];
  arr[i+1] = arr[high];
```

```
    arr[high] = temp;
    return i+1;
}
void sort(int arr[], int low, int high){
    if (low < high){
        /* pi is partitioning index, arr[pi] is
          now at right place */
        int pi = partition(arr, low, high);
        // Recursively sort elements before
        // partition and after partition
        sort(arr, low, pi-1);
        sort(arr, pi+1, high);
    }
    display(high+1,arr);
}
int linearSearch(){
    int size;
    cout<<"Enter the size: ";
```

```
    cin>>size;
    int list[size];
    cout<<"Enter the sorted list  "<<endl;
    for(int i=0;i<size;i++)
        cin>>list[i];
    int num;
    cout<<"Enter the number to be
searched: ";
    cin>>num;
    for(int i=0;i<size;i++){
        if(list[i]==num) {
            return i+1 ;
        }
    }
    return -1;
}
int binarySearch(){
    int size;
```

```
   cout<<"Enter the size: ";
   cin>>size;
   int list[size];
   cout<<"Enter the sorted list  "<<endl;
   for(int i=0;i<size;i++)
      cin>>list[i];
   int num;
   cout<<"Enter the number to be
searched: ";
   cin>>num;
  return BinarySearch(list, 0, size - 1,
num);
}
int BinarySearch(int arr[], int l, int r, int
x){
   if (r>=l){
      int mid = l + (r - l)/2;
      // If the element is present at the
```

```
        // middle itself
        if (arr[mid] == x)
            return mid;
        // If element is smaller than mid,
then
        // it can only be present in left
subarray
        if (arr[mid] > x)
            return BinarySearch(arr, l, mid-1,
x);
        // Else the element can only be
present
        // in right subarray
        return BinarySearch(arr, mid+1, r,
x);
    }
    // We reach here when element is not
present
```

```
    //  in array
    return -1;
}
```