Question:find the index of element using linear and binary search
solution:

```
#include<iostream>
#include<algorithm>
using namespace std;
class SearchingAlgo
{
   public:
   int size;
   int target;
   int *arr;
   bool sorted;
   SearchingAlgo(int size,int target)
   {
      this->size=size;
      this->target=target;
      arr=new int[size];
      cout<<endl<<"Enter the array ";
      for(int i=0;i<size;i++)
      {
         cin>>arr[i];
      }
      sorted=is_sorted(arr,arr+size);

      if (sorted)
      {
         cout<<"Array is sorted"<<endl;
      }
      else
      {
         cout<<"Array is not sorted"<<endl;
      }
      sort(arr,arr+size);
   }
   int linearSearchIterative(int size,int target)
   {
      int i=0;
      for(int i=0;i<size;i++)
      {
         if(arr[i]==target)
         {
            return i;
         }
      }
      return -1;
   }
   int linearSearchRecursive(int size,int target)
   {
      static int i=0;
      if(arr[i]==target)
      {
         return i;
```

```
    }
    if(i>size)
    {
        return -1;
    }
    i++;
    return linearSearchRecursive(size,target);
}
int binarySearchIterative(int start,int size,int target)
{
    int mid;
    mid=(start+size)/2;
    while(start<=size)
    {
        if(arr[mid]==target)
        {
            return mid;
        }
        else
        {
            if(target<arr[mid])
            {
                size=mid-1;
            }
            if(target>arr[mid])
            {
                start=mid+1;
            }
        }
        mid=(start+size)/2;
    }
    return -1;
}
int binarySearchRecursive(int start,int size,int target)
{
    int mid=(start+size)/2;
    if(arr[mid]==target)
    {
        return mid;
    }
    if(start==size)
    {
        return -1;
    }
    if(target>arr[mid])
    {
        return binarySearchRecursive(mid+1,size,target);
    }
    if(target<arr[mid])
    {
        return binarySearchRecursive(start,mid,target);
    }
```

```
        return mid;
    }
};
int main()
{
    int size;
    cout<<"Enter the size ";
    cin>>size;
    int target;
    cout<<endl<<"Enter the target that you want to find ";
    cin>>target;
    SearchingAlgo s(size,target);
    int index;
    int which;
    cout<<endl<<"Enter 1 for linearSearchIterative"<<endl;;
    cout<<"Enter 2 for linearSearchRecursive"<<endl;
    cout<<"Enter 3 for binarySearchIterative"<<endl;
    cout<<"Enter 4 for binarySearchRecursive"<<endl;
    cin>>which;
    switch(which)
    {
        case 1:
        {
            int index=s.linearSearchIterative(size,target);
            cout<<"The index is "<<index;
            break;
        }
        case 2:
        {
            index=s.linearSearchRecursive(size-1,target);
            cout<<"The index is "<<index;
            break;
        }
        case 3:
        {
            index=s.binarySearchIterative(0,size-1,target);
            cout<<"The index is "<<index;
            break;
        }
        case 4:
        {
            index=s.binarySearchRecursive(0,size-1,target);
            cout<<"The index is "<<index;
            break;
        }
        defaut:
        {
            cout<<"This is wrong choice "<<endl;
        }
    }
}
```

output:
//linearSearchIterative
Enter the size 5

Enter the target that you want to find 3

Enter the array 3 1 2 5 4
Array is not sorted

Enter 1 for linearSearchIterative
Enter 2 for linearSearchRecursive
Enter 3 for binarySearchIterative
Enter 4 for binarySearchRecursive
1
The index is 0
//linearSearchRecursive
Enter the size 5

Enter the target that you want to find 6

Enter the array 5 2 1 6 4
Array is not sorted

Enter 1 for linearSearchIterative
Enter 2 for linearSearchRecursive
Enter 3 for binarySearchIterative
Enter 4 for binarySearchRecursive
2
The index is 3
//binarySearchIterative
Enter the size 5

Enter the target that you want to find 3

Enter the array 3 1 2 5 4
Array is not sorted

Enter 1 for linearSearchIterative
Enter 2 for linearSearchRecursive
Enter 3 for binarySearchIterative
Enter 4 for binarySearchRecursive
3
The index is 2
//binarySearchRecursive
Enter the size 5

Enter the target that you want to find 6

Enter the array 5 2 1 6 4
Array is not sorted

Enter 1 for linearSearchIterative

Enter 2 for linearSearchRecursive
Enter 3 for binarySearchIterative
Enter 4 for binarySearchRecursive
4
The index is 3