using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BT3

{

internal class Program

{

public class IntArray

{

private int[] arr;

public int[] Arr

{

get { return arr; }

set { arr = value; }

}

public IntArray()

{

arr = new int[0];

}

public IntArray(int k)

{

Random rand = new Random();

arr = Enumerable.Range(1, k).Select(\_ => rand.Next(1, 201)).ToArray();

}

public IntArray(int[] a)

{

arr = new int[a.Length];

Array.Copy(a, arr, a.Length);

}

public IntArray(IntArray obj)

{

arr = new int[obj.arr.Length];

Array.Copy(obj.arr, arr, obj.arr.Length);

}

public void Input()

{

Console.Write("Enter the size of the array: ");

int n = Convert.ToInt32(Console.ReadLine());

arr = new int[n];

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

{

Console.Write($"Enter element {i + 1}: ");

arr[i] = Convert.ToInt32(Console.ReadLine());

}

}

public void Output()

{

foreach (var item in arr)

Console.Write($"{item} ");

Console.WriteLine();

}

public int SequentialSearch(int x)

{

for (int i = 0; i < arr.Length; i++)

if (arr[i] == x)

return i;

return -1;

}

public void HoanVi(ref int a, ref int b)

{

int tam = a;

a = b;

b = tam;

}

public void InterchangeSort()

{

int n = arr.Length;

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

{

for (int j = i + 1; j < n; j++)

{

if (arr[i] > arr[j])

{

HoanVi(ref arr[i], ref arr[j]);

}

}

}

}

public void BubbleSort()

{

int n = arr.Length;

for (int i = 0; i < n-1; i++)

{

bool swapped = false;

for (int j = 0; j < n - i - 1; j++)

{

if (arr[i] > arr[j + 1])

{

HoanVi(ref arr[i], ref arr[j]);

swapped = true;

}

}

if (!swapped)

break;

}

}

public void SelectionSort()

{

int n = arr.Length;

for (int i = 0; i < n - 1; i++)

{

int minIndex = i;

for (int j = i + 1; j < n; j++)

if (arr[j] < arr[minIndex])

minIndex = j;

if (minIndex != i)

{

HoanVi(ref minIndex, ref arr[i]);

}

}

}

public void InsertionSort()

{

int n = arr.Length;

for (int i = 1; i < n; ++i)

{

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key)

{

HoanVi(ref key, ref arr[i]);

j = j - 1;

}

arr[j + 1] = key;

}

}

public void QuickSort()

{

QuickSortHelper(0, arr.Length - 1);

}

private void QuickSortHelper(int low, int high)

{

if (low < high)

{

int pi = Partition(low, high);

QuickSortHelper(low, pi - 1);

QuickSortHelper(pi + 1, high);

}

}

private int Partition(int low, int high)

{

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++)

{

if (arr[j] < pivot)

{

i++;

HoanVi(ref arr[i], ref arr[j]);

}

}

HoanVi(ref arr[i + 1], ref arr[high]);

return (i + 1);

}

}

static void Main(string[] args)

{

Console.Write("Enter the size of the array: ");

int k = Convert.ToInt32(Console.ReadLine());

IntArray objA = new IntArray(k);

Console.WriteLine("The random values in objA are: ");

objA.Output();

Console.Write("Enter the value to find: ");

int x = Convert.ToInt32(Console.ReadLine());

int index = objA.SequentialSearch(x);

if (index != -1)

Console.WriteLine($"Found {x} at index {index} in objA.");

else

Console.WriteLine($"{x} is not found in objA.");

Console.WriteLine("Mang sau khi sap xep Interchangesort");

objA.InterchangeSort();

objA.Output();

Console.WriteLine("Mang sau khi sap xep BobbleSort");

objA.BubbleSort();

objA.Output();

Console.WriteLine("Mang sau khi sap xep SelectionSortt");

objA.SelectionSort();

objA.Output();

Console.WriteLine("Mang sau khi sap xep InsertionSort");

objA.InsertionSort();

objA.Output();

Console.WriteLine("Mang sau khi sap xep QuickSort");

objA.QuickSort();

objA.Output();

}

}

}