DSA-ASSIGNMENT-2

Searching

Q1. Given an array. Find the number X in the array. If the element is present, return the index of the element, else print “Element not found in array”. Input the size of array, array from user and the element X from user. Use Linear Search to find the element.

Sol:

```java
import java.util.Scanner;
public class LinearSearch {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number of elements: ");
        int n=sc.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter array elements: ");
        for (int i=0;i<n;i++)
        {
            arr[i]= sc.nextInt();
        }
        System.out.println("Enter the element you want to search: ");
        int x= sc.nextInt();
        int idx=-1;
        for (int i=0;i< arr.length;i++)
        {
            if (arr[i]==x)
            {
                idx=i;
               break;
            }
        }
        if(idx==-1){
            System.out.println("Element not found!!");
        }
        else
            System.out.println("Element Found at: "+idx);
    }
}
```

Q2. Given an array and an integer “target”, return the last occurrence of “target” in the array. If the target is
not present return -1.
Input 1: arr = [1 1 1 2 3 4 4 5 6 6 6 6 , target = 4
Output 1: 6
Input 2: arr = [2 2 2 6 6 18 29 30 30 30], target = 15
Output 2: -1
Sol:

```java
import java.util.Scanner;
public class SearchElement {
    public static int searchElement(int arr[], int target) {
        int low = 0;
        int high = arr.length - 1;
        int answer = -1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == target) {
                answer = mid;
                low = mid + 1;
            } else if (target > arr[mid]) {
                high = mid - 1;
            } else
```

```
                low = mid + 1;
        }
        return answer;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements: ");
        int n = sc.nextInt();
        int arr[] = new int[n];
        System.out.println("Enter array elements: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter the element you want to search: ");
        int target = sc.nextInt();
        int answer=searchElement(arr,target);
        if(answer==-1)
            System.out.println(answer);
        else
            System.out.println("The last occurrence of the target element is at: 
"+answer);
    }
}
```

Q3. Given a sorted binary array, efficiently count the total number of 1's in it.
Input 1: arr = [0 0 0 0 1 1 1 1 1 1]
Output 1: 6
Input 2: arr = [0 0 0 0 0 1 1]
Output 2: 2
Sol:

```
import java.util.Scanner;
public class CountOnes {
    public static int counter(int arr[]) {
        int low = 0;
        int high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == 0) {
                low = mid + 1;
            }
            else if (arr[mid] > 0) {
                high = mid - 1;
            }
        }
        return (arr.length - low);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements: ");
        int n = sc.nextInt();
        int arr[] = new int[n];
        System.out.println("Enter array elements: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        int answer=counter(arr);
        if(answer==-1)
            System.out.println(answer);
        else
            System.out.println("Total no. of 1s are: "+answer);
    }
}
```

Q4. Given a sorted integer array containing duplicates, count occurrences of a given number. If the element is not found in the array, report that as well.
Input: nums[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 5
Output: Target 5 occurs 3 times
Input: nums[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 6
Output: Target 6 occurs 2 times
Sol:

```java
import java.util.Scanner;
import java.util.Scanner;
public class TargetCount {
    public static int searchElement2(int arr[], int target) {
        int low = 0;
        int high = arr.length - 1;
        int answer=-1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == target) {
                answer=mid;
                low = mid + 1;
            } else if (arr[mid]>target) {
                high = mid - 1;
            } else
                low = mid + 1;
        }
        return answer;
    }
    public static int searchElement1(int arr[], int target) {
        int low = 0;
        int high = arr.length - 1;
        int answer=-1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == target) {
                answer=mid;
                high = mid - 1;
            } else if (arr[mid]>target) {
                high = mid - 1;
            } else
                low = mid + 1;
        }
        return answer;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements: ");
        int n = sc.nextInt();
        int arr[] = new int[n];
        System.out.println("Enter array elements: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter the element you want to search: ");
        int target = sc.nextInt();
        int first=searchElement1(arr,target);
        int last=searchElement2(arr,target);
        if (first==last && first==-1) System.out.println("Not Found");
        else System.out.println("The total no. of target element are: "+(last-
first+1));
    }
}
```

Q5: Given a positive integer num, return true if num is a perfect square or false otherwise.
A perfect square is an integer that is the square of an integer. In other words, it is the product of some integer with itself.
Example 1:
Input: num = 16
Output: true
Explanation: We return true because 4 * 4 = 16 and 4 is an integer.
Example 2:
Input: num = 14
Output: false
Explanation: We return false because 3.742 * 3.742 = 14 and 3.742 is not an integer.
Sol:

```
import java.util.Scanner;
public class PerfectSquare {
    public static boolean isPerfectSquare(int num){
        long low=0;
        long high=num/2;
        while (low<=high)
        {
            long mid=low+(high-low)/2;
            if(mid*mid>num)high=mid-1;
            else if (mid*mid<num)low=mid+1;
            else return true;
        }
        return false;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number: ");
        int num=sc.nextInt();
        System.out.println("The number is perfect square: "+isPerfectSquare(num));
    }
}
```

<u>Assignment- sorting</u>

Q1. Write a program to sort an array in descending order using bubble sort.
Input Array {3,5,1,6,0}
Output Array: {6, 5, 3, 1, 0}
Sol:

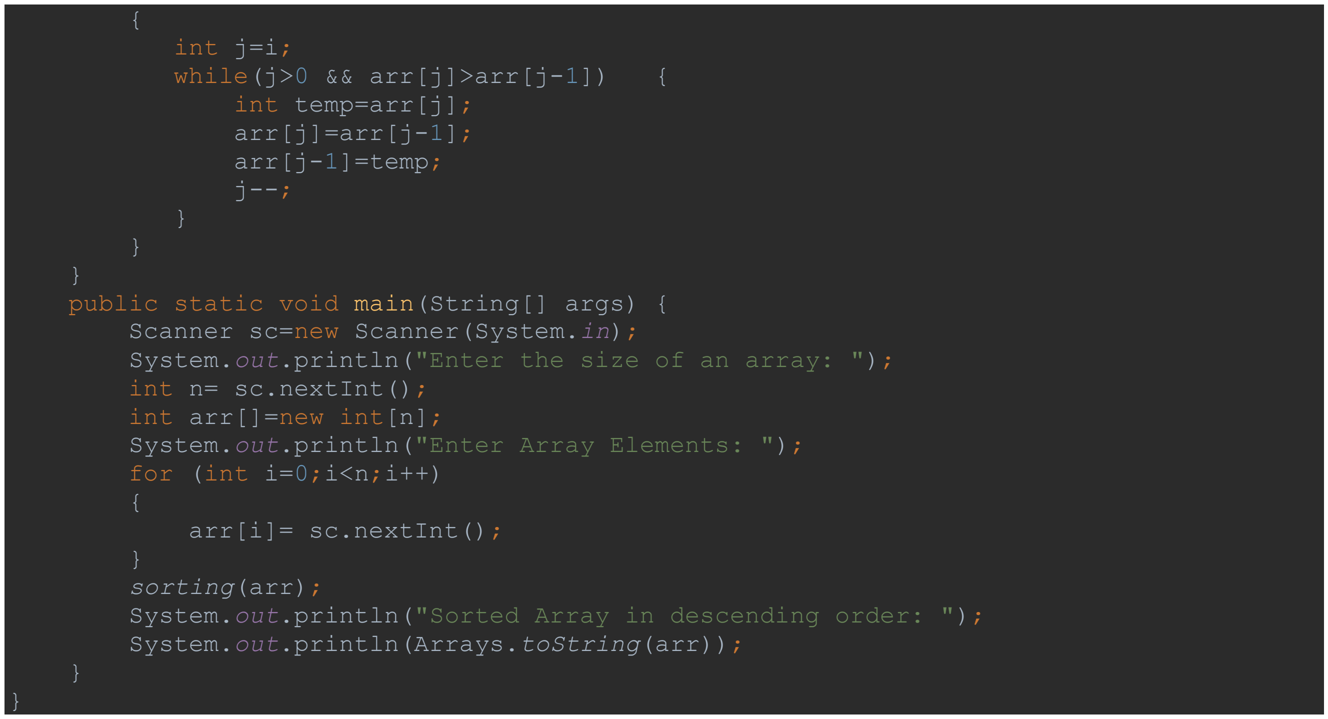
```
import java.util.Arrays;
import java.util.Scanner;
public class Sort {
    public static void sorting(int[] arr){
        for (int i=0;i< arr.length;i++)
        {
            boolean swapped=false;
            for (int j=0;j< arr.length-i-1;j++)
            {
                if(arr[j+1]>arr[j]){
                    int temp=arr[j+1];
                    arr[j+1]=arr[j];
                    arr[j]=temp;
                    swapped=true;
                }
            }
            if (!swapped)break;
        }
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the size of an array: ");
        int n= sc.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter Array Elements: ");
        for (int i=0;i<n;i++)
        {
            arr[i]= sc.nextInt();
        }
        sorting(arr);
        System.out.println("Sorted Array in descending order: ");
        System.out.println(Arrays.toString(arr));
    }
}
```

Q2. WAP to sort an array in descending order using selection sort
Input Array {3,5,1,6,0}
Output Array: {6, 5, 3, 1, 0}
Sol:

```
import java.util.Arrays;
import java.util.Scanner;
public class Sort2 {
    public static void sorting(int[] arr){
        for (int i=0;i< arr.length;i++)
        {
            int max_idx=i;
            for (int j=i+1;j< arr.length;j++)
            {
                if (arr[j]>arr[max_idx])
                {
                    max_idx=j;
                }
            }
            if(max_idx!=i)
            {
                int temp=arr[max_idx];
                arr[max_idx]=arr[i];
                arr[i]=temp;
            }
        }
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the size of an array: ");
        int n= sc.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter Array Elements: ");
        for (int i=0;i<n;i++)
        {
            arr[i]= sc.nextInt();
        }
        sorting(arr);
        System.out.println("Sorted Array in descending order: ");
        System.out.println(Arrays.toString(arr));
    }
}
```

Q3. WAP to sort an array in decreasing order using insertion sort
Input Array {3,5,1,6,0}
Output Array: {6, 5, 3, 1, 0}
Sol:

```
import java.util.Scanner;
import java.util.Arrays;
public class Sort3 {
    public static void sorting(int[] arr){
        for (int i=0;i< arr.length;i++)
```



```
        {
            int j=i;
            while(j>0 && arr[j]>arr[j-1])    {
                int temp=arr[j];
                arr[j]=arr[j-1];
                arr[j-1]=temp;
                j--;
            }
        }
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the size of an array: ");
        int n= sc.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter Array Elements: ");
        for (int i=0;i<n;i++)
        {
            arr[i]= sc.nextInt();
        }
        sorting(arr);
        System.out.println("Sorted Array in descending order: ");
        System.out.println(Arrays.toString(arr));
    }
}
```

Q4. Find out how many pass would be required to sort the following array in decreasing order using bubble sort
Input Array {3,5,1,6,0}
Sol:
Array= {3,5,1,6,0}
In first iteration= {5,3,6,1,0}
In second iteration= {5,6,3,1,0}
In third iteration= {6,5,3,1,0}
Hence, 3 passes are required to sort the array in decreasing order.

Q5. Find out the number of iterations to sort the array in descending order using selection sort.
Input Array {3,5,1,6,0}
Sol:
Array= {3,5,1,6,0}
In first iteration= {6,5,1,3,0}
In second iteration= {6,5,1,3,0}
In third iteration= {6,5,3,1,0}
Hence, 3 iterations are required to sort the array in descending order.

Assignment- Number system

Problem 1: given a number, print its binary representation. [easy]
Input 1: number = 5
Output 1: 101
Input 2: number = 10
Output 2: 1010
Sol:
2|5
2|2 ->1
2|1 ->0
|1
(5) = 101

2|10
2|5 ->0
2|2 ->1
2|1 ->0
|1
(10) = 1010

Problem 2: given a number 'n', predict whether it is a power of two or not. [medium]
Input 1: n = 15
Output 1: False
Input 2: n = 32
Output 2: True
Sol:
Converting 32 into binary
(32)=100000
It has only 1 set bit, hence it is a power of 2.

Q3. Problem 1: Given a number. Using bit manipulation, check whether it is odd or even.
Input 8, Even
3, False
Sol:

```
import java.util.Scanner;
public class OddEven {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number: ");
        int num= sc.nextInt();
        if((num & 1)==1) System.out.println("The given number is odd");
        else System.out.println("The given number is even");
    }
}
```

Q4. Given a number, count the number of set bits in that number without using an extra space.
Note : bit '1' is also known as set bit.
Sol:

```
import java.util.Scanner;
public class SetCount {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the Number");
        int num = sc.nextInt();
        int count=0;
        while(num>0)
        {
            count += num & 1;
            num >>= 1;
        }
        System.out.println("The total no. of Set are: "+count);
    }
}
```

Q5. Given an integer array, duplicates are present in it in a way that all duplicates appear an even number of times except one which appears an odd number of times. Find that odd appearing element in linear time and without using any extra memory.
For example,
Input : arr[] = [4, 3, 6, 2, 6, 4, 2, 3, 4, 3, 3]
Output : The odd occurring element is 4.
Sol:

```
import java.util.Scanner;
public class OddNoElement {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter size of array: ");
        int n= sc.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter array elements:");
        for (int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }
        int xor=0;
        for (int i:arr) {
            xor=xor^i;
        }
        System.out.println("The odd occurring element is: "+xor);
    }
}
```