Batch – T5
Assignment No. - 2
Title - Searching Algorithm
Student Name - Sharaneshwar Bharat Punjal
Student PRN - 23520011

1) You are an IT company's manager. Based on their performance over the last N working days, you must rate your employee. You are given an array of N integers called workload, where workload[i] represents the number of hours an employee worked on an ith day. The employee must be evaluated using the following criteria:
Rating = the maximum number of consecutive working days when the employee has worked more than 6 hours.
You are given an integer N where N represents the number of working days. You are given an integer array workload where workload[i] represents the number of hours an employee worked on an ith day.
Task: Determine the employee rating

```java
import java.util.Scanner;

public class Q1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] workload = new int[n];
        for (int i = 0; i < workload.length; i++)
            workload[i] = sc.nextInt();
        sc.close();

        System.out.println(getRating(workload, n));
    }

    private static int getRating(int[] workload, int N) {
        int rating = Integer.MIN_VALUE;
        int cnt = 0;

        for (int i : workload) {
            if (i > 6)
                cnt++;
            else {
                rating = Math.max(rating, cnt);
                cnt = 0;
            }
        }
        rating = Math.max(rating, cnt);
        return rating;
    }
}
```

Test Case 1:
7
8 7 5 6 10 9 2
Output:
2

Test Case 2:
5
7 8 9 10 11
Output:
5

Test Case 3:
4
3 4 5 6
Output:
0

Test Case 4:
20
5 6 7 8 2 3 4 5 9 10 11 12 1 2 3 6 7 8 1 2
Output:
4

2) You have N boxes numbered 1 through N and K candies numbered 1 through K. You put the candies in the boxes in the following order:
- first candy in the first box,
- second candy in the second box,
- ....
- ....
- so up to N-th candy in the Nth box,
- the next candy in (N - 1)-th box,
- the next candy in (N - 2)-th box
- ....
- ....
- and so on up to the first box,
- then the next candy in the second box
- ..... and so on until there is no candy left.

So you put the candies in the boxes in the following order:
Find the index of the box where you put the K-th candy.

```java
import java.util.Scanner;

public class Q2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();

        while (T-- > 0) {
            int N = sc.nextInt();
```

```
            int K = sc.nextInt();
            System.out.println(findBoxIndex(N, K));
        }

        sc.close();
    }

    private static int findBoxIndex(int N, int K) {
        int cycleLength = 2 * (N - 1);

        int pos = (K - 1) % cycleLength;

        if (pos < N)
            return pos + 1;
        else
            return N - (pos - N + 1);
    }
}
```

Test Case 1:

```
4
5 7
4 10
6 20
10 25
```

Output:

```
3
2
6
6
```

Test Case 2:

```
3
2 5
3 8
7 14
```

Output:

```
2
2
7
```

Test Case 3:

```
4
3 7
5 13
6 18
8 24
```

Output:

```
1
3
6
```

8

Test Case 4:

3
4 100
7 150
10 200

Output:

4
6
10

3) Implement and Explain Tower of Hanoi algorithm.

```
import java.util.Scanner;

public class Q3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.close();
        towerOfHanoi(n, 'A', 'C', 'B');
    }

    private static void towerOfHanoi(int n, char from, char to, char aux) {
        if (n == 1) {
            System.out.println("Move disk 1 from rod " + from + " to rod " + to);
            return;
        }

        towerOfHanoi(n - 1, from, aux, to);
        System.out.println("Move disk " + n + " from rod " + from + " to rod " + to);
        towerOfHanoi(n - 1, aux, to, from);
    }
}
```

Test Case 1:

3

Output:

Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C

Test Case 2:

2

Output:

Move disk 1 from rod A to rod B

```
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
```

Test Case 3:

```
4
```

Output:

```
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 3 from rod A to rod B
Move disk 1 from rod C to rod A
Move disk 2 from rod C to rod B
Move disk 1 from rod A to rod B
Move disk 4 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 2 from rod B to rod A
Move disk 1 from rod C to rod A
Move disk 3 from rod B to rod C
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
```

Test Case 4:

```
5
```

Output:

```
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
Move disk 4 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 2 from rod C to rod A
Move disk 1 from rod B to rod A
Move disk 3 from rod C to rod B
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 5 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
Move disk 3 from rod B to rod A
Move disk 1 from rod C to rod B
Move disk 2 from rod C to rod A
Move disk 1 from rod B to rod A
Move disk 4 from rod B to rod C
Move disk 1 from rod A to rod C
```

```
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

4) There is a frog initially placed at the origin of the coordinate plane. In exactly 1 second, the frog can either move up 1 unit, move right 1 unit, or stay still. In other words, from position (x, y), the frog can spend 1 second to move to:

- (X + 1, Y)
- (X, Y + 1)
- (X, Y)

After T seconds, a villager who sees the frog reports that the frog lies on or inside a square of side-length s with coordinates (X, Y), (X + S, Y), (X, Y + S), (X + S, Y + S). Calculate how many points with integer coordinates on or inside this square could be the frog's position after exactly T seconds

Input Format:

The first and only line of input contains four space-separated integers: X, Y, S, and T.

Output Format:

Print the number of points with integer coordinates that could be the frog's position after T seconds.

```java
import java.util.Scanner;

public class Q4 {
    public static void main(String args[] ) throws Exception {
        Scanner sc = new Scanner(System.in);
        int X = sc.nextInt();
        int Y = sc.nextInt();
        int s = sc.nextInt();
        int T = sc.nextInt();

        int count = 0;
        for (int i = X; i <= X + s; i++)
            for (int j = Y ; j <= Y + s; j++)
                if (i + j <= T)
                    count++;

        System.out.println(count);

        sc.close();
    }
}
```

Test Case 1:

```
1 1 2 5
```

Output:

```
8
```

Test Case 2:

0 0 1 2

Output:

4

Test Case 3:

2 2 3 6

Output:

6

Test Case 4:

2 60 95 116

Output:

1540

5) Implement linear Search Algorithm.

```
import java.util.Scanner;

public class Q5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();

        int key = sc.nextInt();
        sc.close();

        int index = linearSearch(arr, key);
        if (index == -1) {
            System.out.println("Element not found");
        } else {
            System.out.println("Element found at index: " + index);
        }
    }

    public static int linearSearch(int[] arr, int key) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                return i;
            }
        }
        return -1;
    }
}
```

Test Case 1:

25

10 22 35 47 50 65 78 89 90 102 113 124 135 146 157 168 179 190 201 212 223 234 245 256 267

124
Output:
Element found at index: 11

Test Case 2:
30
5 17 29 41 53 65 77 89 101 113 125 137 149 161 173 185 197 209 221 233 245 257 269 281 293 305 317 329 341 353
305
Output:
Element found at index: 25

Test Case 3:
22
8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128 136 144 152 160 168 176
474
Output:
Element not found

Test Case 4:
24
7 14 21 28 35 42 49 56 63 70 77 84 91 98 105 112 119 126 133 140 147 154 161 168
140
Output:
Element found at index: 20

6) Implement Binary Search algorithm.

```
import java.util.Scanner;

public class Q6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();

        int key = sc.nextInt();
        sc.close();

        int index = binarySearch(arr, key);

        if (index == -1) {
            System.out.println("Element not found");
        } else {
            System.out.println("Element found at index: " + index);
        }
    }

    public static int binarySearch(int[] arr, int key) {
```

```
        int low = 0;
        int high = arr.length - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (arr[mid] == key) {
                return mid;
            }

            if (arr[mid] < key) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return -1;
    }
}
```

Test Case 1:

```
15
3 7 11 16 20 25 30 35 40 45 50 55 60 65 70
55
```

Output:

```
Element found at index: 11
```

Test Case 2:

```
20
-10 -5 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85
5
```

Output:

```
Element found at index: 3
```

Test Case 3:

```
25
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900
2000 2100 2200 2300 2400 2500
1950
```

Output:

```
Element not found
```

Test Case 4:

```
18
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
28
```

Output:

```
Element found at index: 13
```