# Week 1 Advanced Coding Tasks

SAI SREEVATSAV K

HU21CSEN0101138

#LeetCode: Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

```
import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Solution {

  public int[] twoSum(int[] nums, int target) {

    Map<Integer, Integer> complementMap = new HashMap<>();


    for (int i = 0; i < nums.length; i++) {

      int complement = target - nums[i];

      if (complementMap.containsKey(complement)) {

        return new int[]{complementMap.get(complement), i};

      }


      complementMap.put(nums[i], i);

    }

    throw new IllegalArgumentException("No valid solution exists.");

  }


  public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the array elements:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        System.out.print("Enter the target value: ");
        int target = scanner.nextInt();

        Solution solution = new Solution();
        int[] result = solution.twoSum(nums, target);
        System.out.println("Indices: [" + result[0] + ", " + result[1] + "]");
    }
}
```

#HackerRank: Diagonal Difference: Calculate the absolute difference between the sums of the diagonals in a square matrix.

```
import java.util.Scanner;
public class DiagonalDifference {
    public static int diagonalDifference(int[][] arr) {
        int n = arr.length;
        int primaryDiagonalSum = 0;
        int secondaryDiagonalSum = 0;
        for (int i = 0; i < n; i++) {
            primaryDiagonalSum += arr[i][i];
            secondaryDiagonalSum += arr[i][n - 1 - i];
        }

        return Math.abs(primaryDiagonalSum - secondaryDiagonalSum);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
          int n = scanner.nextInt();
          int[][] arr = new int[n][n];
        for (int i = 0; i < n; i++) {
           for (int j = 0; j < n; j++) {
               arr[i][j] = scanner.nextInt();
           }
        }
        int result = diagonalDifference(arr);
        System.out.println("" + result);
        scanner.close();
    }
}
```

#CodeChef: Life, the Universe, and Everything: Write a program that reads numbers from input and stops processing input after reading the number 42.

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int[] num = new int[10];
        int i = 0;
        Scanner scanner = new Scanner(System.in);
        while (true) {
            if (i >= 10) break;
            num[i] = scanner.nextInt();
            if (num[i] == 42) break;
            i++;
        }
        for (int j = 0; j < i; j++) {
            System.out.println(num[j]);
        }
        scanner.close();
    }
}
```

#Codeforces: Watermelon: Determine if a watermelon can be split into two parts, each of which weighs an even number of kilos.

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        if (((n - 2) % 2 == 0) && (n - 2 > 0)) {
            System.out.print("YES");
        } else {
            System.out.print("NO");
        }
        System.out.println();
        scanner.close();
    }
}
```

#GeeksforGeeks: Reverse Array in Groups: Given an array, reverse every sub-array formed by consecutive k elements.

```
class Main {
  static void reverse(int arr[], int n, int k)
  {
    for (int i = 0; i < n; i += k)
    {
      int left = i;
      int right = Math.min(i + k - 1, n - 1);
      int temp;
      while (left < right)
      {
        temp=arr[left];
        arr[left]=arr[right];
        arr[right]=temp;
        left+=1;
        right-=1;
      }
    }
  }
  public static void main(String[] args)
  {

    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int k = 3;
```

```
        int n = arr.length;
        reverse(arr, n, k);
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
    }
}
```

#AtCoder: Product: Find the product of two integers.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int product = a * b;
        if (product % 2 == 0) {
            System.out.println("Even");
        } else {
            System.out.println("Odd");
        }

        scanner.close();
    }
}
```

#Exercism: Hamming: Calculate the Hamming Distance between two DNA strands.

```
public class Main {
  public static int calculateHammingDistance(String strand1, String strand2) {
    if (strand1.length() != strand2.length()) {
      throw new IllegalArgumentException("Strands must be of equal length.");
    }
    int distance = 0;
    for (int i = 0; i < strand1.length(); i++) {
      if (strand1.charAt(i) != strand2.charAt(i)) {
        distance++;
      }
    }

    return distance;
  }

  public static void main(String[] args) {
    String strand1 = "GAGCCTACTAACGGGAT";
    String strand2 = "CATCGTAATGACGGCCT";
    try {
      int distance = calculateHammingDistance(strand1, strand2);
      System.out.println("Hamming Distance: " + distance);
    } catch (IllegalArgumentException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

#TopCoder: SRM 758 Div 2 - Very Easy Problem: Given an integer N, determine if it is possible to create an array of integers that sums to N.

```
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int N = scanner.nextInt();
    if (N > 0) {
      System.out.println("YES");
    } else {
      System.out.println("NO");
    }
    scanner.close();
  }
}
```

#CSES Problem Set: Missing Number: Find the missing number in a list of n integers where one number from 1 to n is missing.

```
import java.util.*;
public class Main {
  static int solve(int N, int[] arr) {
    int XOR = 0;
    for (int i = 0; i < N - 1; i++) {
      XOR ^= arr[i];
      XOR ^= (i + 1);
    }
```

```
        XOR ^= N;
        return XOR;
    }
    public static void main(String[] args) {
        int N = 5;
        int[] arr = {2, 3, 1, 5};
        System.out.println(solve(N, arr));
    }
}
```

#InterviewBit: Find Duplicate in Array: Given a read-only array of n+1 integers between 1 and n, find one duplicate number."

```
import java.util.Scanner;
class Main {
    public static int findDuplicate(int[] A) {
        int n = A.length - 1;
        int slow = A[0];
        int fast = A[A[0]];

        while (slow != fast) {
            slow = A[slow];
            fast = A[A[fast]];
        }
        fast = 0;
        while (slow != fast) {
            slow = A[slow];
            fast = A[fast];
        }

        return slow;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        scanner.nextLine();
        int[] A = new int[n + 1];
        System.out.println((n + 1) + n );
        String[] input = scanner.nextLine().split("\\s+");
        if (input.length != n + 1) {
            System.out.println((n + 1));
            scanner.close();
            return;
        }
        try {
            for (int i = 0; i < n + 1; i++) {
                A[i] = Integer.parseInt(input[i]);
                if (A[i] < 1 || A[i] > n) {
                    throw new IllegalArgumentException(n);
                }
            }
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format.");
            scanner.close();
            return;
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
            scanner.close();
            return;
        }
        int duplicate = findDuplicate(A);
        System.out.println(duplicate);
        scanner.close();
    }
}
```