Problem Solving - V - Arraylist, Control Flow Statements, S
Priority Queue, Linked List, Hashset

1. Magical Pattern

Problem statement :

You have been given an integer 'N'. Your task is to print the Magical Pattern(see examples) for the given 'N'.

Example :

For 'N' : 4

Pattern :

```
4 3 2 1 2 3 4
3 3 2 1 2 3 3
2 2 2 1 2 2 2
1 1 1 1 1 1 1
2 2 2 1 2 2 2
3 3 2 1 2 3 3
4 3 2 1 2 3 4
```

Analysis

- The pattern is symmetric about the middle row and column.
- The middle row and column is the row and column where the value is 1.
- The value of the middle row and column is 'N'.
- Print the pattern.

Code:

```java
package com.hands_on;

import java.util.Scanner;

public class MagicalPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter any number: ");
        int n = sc.nextInt();
        for ( int i = 1 ; i < 2*n; i++) {
            for (int j = 1; j < 2 * n; j++)
                System.out.print(n - Math.max(Math.min(i, 2 * n - i), Math.min(j, 2 * n - j)) + 1 + " ");
            System.out.println();
        }
    }
}
```

Code - using ArrayList:

```java
package com.hands_on;

import java.util.ArrayList;
import java.util.Scanner;
```

```java
public class MagicalPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter any number: ");
        int n = sc.nextInt();
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();
        for(int i=1; i<2*n; i++){
            ArrayList<Integer> temp = new ArrayList<>();
            for(int j=1; j<2*n; j++){
                temp.add(n - Math.max(Math.min(i, 2 * n - i), Math.min(j, 2 * n - j)) + 1);
            }
            list.add(temp);
        }

        for(ArrayList<Integer> i: list){
            for(int j: i){
                System.out.print(j + " ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
Enter any number: 4
4 3 2 1 2 3 4
3 3 2 1 2 3 3
2 2 2 1 2 2 2
1 1 1 1 1 1 1
2 2 2 1 2 2 2
3 3 2 1 2 3 3
4 3 2 1 2 3 4
```

---

2. Interesting Alphabets

Problem statement :

As a part of its competition, the school will conduct a codeathon, Lock the Code, where it has been given a value, and the participants have to decode it. The participar number of rows in the matrix; they need to print the pattern.

Example :

For N=5, Pattern:

```
E
DE
CDE
BCDE
ABCDE
```

Analysis

- The pattern is a right-angled triangle.
- The first row has the last alphabet.
- The second row has the last two alphabets.
- The third row has the last three alphabets.
- The pattern continues till the Nth row.

Code:

```java
package com.hands_on;

import java.util.Scanner;

public class InterestingAlphabets {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter any number: ");
        int n = sc.nextInt();
        for(int i=1; i<n+1; i++){
            for(int j=n-i; j<n; j++ ){
                System.out.print((char)(j+65));
            }
            System.out.println();
        }
    }
}
```

Code - using ArrayList:

```java
package com.hands_on;

import java.util.ArrayList;
import java.util.Scanner;

public class InterestingAlphabetsArrayList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter any number: ");
        int n = sc.nextInt();
        ArrayList<ArrayList<Character>> list = new ArrayList<>();
        for(int i=1; i<n+1; i++){
            ArrayList<Character> temp = new ArrayList<>();
            for(int j=n-i; j<n; j++){
                temp.add((char)(j+65));
            }
            list.add(temp);
        }

        for(ArrayList<Character> i: list){
            for(char j: i){
                System.out.print(j);
            }
            System.out.println();
        }
    }
}
```

Output:

```
Enter any number: 5
E
DE
CDE
BCDE
ABCDE
```

---

3. Word Pattern

Problem statement :

Given a pattern and a string s, find if s follows the same pattern.

Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in s

Analysis

- Split the string s into words.
- If the length of the pattern and the number of words are not equal, return false.
- Create a hashmap to store the mapping between the character in the pattern and the word in the string.
- For each character in the pattern, check if the character is already present in the hashmap.
- If present, check if the word matches the word in the hashmap.
- If not present, check if the word is already present in the hashmap.
- If present, return false.

Code

```java
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

class WordPattern {
    public static boolean wordPattern(String pattern, String s) {
        String[] word = s.split(" ");
        if(word.length != pattern.length()) return false;

        HashMap<Character, String> hashMap = new HashMap<>();
        for(int i=0; i<pattern.length(); i++){
            char c = pattern.charAt(i);

            if(hashMap.containsKey(c)){
                if(!hashMap.get(c).equals(word[i])){
                    return false;
                }
            }else {
                if(hashMap.containsValue(word[i])){
                    return false;
                }
                hashMap.put(c, word[i]);
            }
        }

        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the string - 1: ");
        String pattern = sc.nextLine();
        System.out.print("Enter the string - 2: ");
        String s = sc.nextLine();
        System.out.println(wordPattern(pattern, s));
    }
}
```

Output:

```
Enter the string - 1: abba
Enter the string - 2: dog cat cat dog
true
```

---

4. Pascal's Triangle II

Problem statement :

Given an integer rowIndex, return the rowIndexth (0-indexed) row of the Pascal's triangle.

In Pascal's triangle, each number is the sum of the two numbers directly above it as shown

Analysis

- The first and the last element of each row is 1.
- The value at the ith index of the row is the product of the value at the (i-1)th index of the row and the value of (rowIndex-i+1) divided by i.
- Return the row.

Code

```java
package com.hands_on;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class PascalsTriangleII {
    public static List<Integer> getRow(int rowIndex) {
        List<Integer> row = new ArrayList<>();
        row.add(1);
        for(int i=1; i<=rowIndex; i++){
            row.add((int)((long)row.get(i-1)*(rowIndex-i+1)/i));
        }
        return row;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter any number: ");
        int rowIndex = sc.nextInt();
        System.out.println(getRow(rowIndex));
    }
}
```

Output:

```
Enter any number: 3
[1, 3, 3, 1]
```

---

5. Intersection of Two Arrays

Problem statement :

Given two integer arrays nums1 and nums2, return an array of their intersection

Each element in the result must be unique and you may return the result in any order.

Analysis

- Create a hashset to store the unique elements of the first array.
- Create a hashset to store the unique elements of the second array.
- Create a hashset to store the intersection of the two arrays.
- Return the intersection.

Code

```java
package com.hands_on;

import java.util.HashSet;
import java.util.Scanner;

public class IntersectionOfTwoArrays {
    public static int[] intersection(int[] nums1, int[] nums2) {
        HashSet<Integer> set1 = new HashSet<>();
        HashSet<Integer> set2 = new HashSet<>();

        for(int i: nums1){
            set1.add(i);
        }

        for(int i: nums2){
            set2.add(i);
        }

        set1.retainAll(set2);

        int[] result = new int[set1.size()];
        int i=0;
        for(int num: set1){
            result[i++] = num;
        }

        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the first array: ");
        int n = sc.nextInt();
        int[] nums1 = new int[n];
        System.out.println("Enter the elements of the first array: ");
        for(int i=0; i<n; i++){
            nums1[i] = sc.nextInt();
        }

        System.out.print("Enter the size of the second array: ");
        int m = sc.nextInt();
        int[] nums2 = new int[m];
        System.out.println("Enter the elements of the second array: ");
        for(int i=0; i<m; i++){
            nums2[i] = sc.nextInt();
        }

        int[] result = intersection(nums1, nums2);
        System.out.print("Intersection of the two arrays: ");
        for(int i: result){
            System.out.print(i + " ");
        }
```

```
    }
}
```

Output

```
Enter the size of the first array: 4
Enter the elements of the first array:
1 2 2 1
Enter the size of the second array: 2
Enter the elements of the second array:
2 2
Intersection of the two arrays: 2
```

---

6. Find the Difference of Two Arrays

Problem statement :

Given two 0-indexed integer arrays nums1 and nums2, return a list answer of size 2 where:

- answer[0] is a list of all distinct integers in nums1 which are not present in nums2.
- answer[1] is a list of all distinct integers in nums2 which are not present in nums1.

Note that the integers in the lists may be returned in any order.

Analysis

- Create a hashset to store the unique elements of the first array.
- Create a hashset to store the unique elements of the second array.
- Create a hashset to store the difference of the two arrays.
- Return the difference.

Code

```java
package com.hands_on;

import java.util.HashSet;
import java.util.Scanner;

public class FindTheDifferenceOfTwoArrays {
    public static int[][] findDifference(int[] nums1, int[] nums2) {
        HashSet<Integer> set1 = new HashSet<>();
        HashSet<Integer> set2 = new HashSet<>();

        for(int i: nums1){
            set1.add(i);
        }

        for(int i: nums2){
            set2.add(i);
        }

        HashSet<Integer> diff1 = new HashSet<>(set1);
        HashSet<Integer> diff2 = new HashSet<>(set2);

        diff1.removeAll(set2);
        diff2.removeAll(set1);

        int[][] result = new int[2][];
        result[0] = diff1.stream().mapToInt(Integer::intValue).toArray();
```

```
        result[1] = diff2.stream().mapToInt(Integer::intValue).toArray();

        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the first array: ");
        int n = sc.nextInt();
        int[] nums1 = new int[n];
        System.out.println("Enter the elements of the first array: ");
        for(int i=0; i<n; i++){
            nums1[i] = sc.nextInt();
        }

        System.out.print("Enter the size of the second array: ");
        int m = sc.nextInt();
        int[] nums2 = new int[m];
        System.out.println("Enter the elements of the second array: ");
        for(int i=0; i<m; i++){
            nums2[i] = sc.nextInt();
        }

        int[][] result = findDifference(nums1, nums2);
        System.out.println("Difference of the two arrays: ");
        for(int i=0; i<2; i++){
            for(int j: result[i]){
                System.out.print(j + " ");
            }
            System.out.println();
        }
    }
}
```

Output

```
Enter the size of the first array: 3
Enter the elements of the first array:
1 2 4
Enter the size of the second array: 3
Enter the elements of the second array:
2 4 6
Difference of the two arrays:
1
6
```

---

7. Maximum Product of Two Elements in an Array

Problem statement :

Given the array of integers nums, you will choose two different indices i and j of that array. Return the maximum value of (nums[i]-1)*(nums[j]-1).

Analysis

- Find the two maximum elements in the array.
- Return the product of the two maximum elements minus 1.

Code

```java
package com.hands_on;

import java.util.Scanner;

public class MaximumProductOfTwoElementsInAnArray {
    public static int maxProduct(int[] nums) {
        int max1 = Integer.MIN_VALUE;
        int max2 = Integer.MIN_VALUE;

        for(int i: nums){
            if(i > max1){
                max2 = max1;
                max1 = i;
            }else if(i > max2){
                max2 = i;
            }
        }

        return (max1-1)*(max2-1);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            nums[i] = sc.nextInt();
        }

        System.out.println("Maximum product of two elements in the array: " + maxProduct(nums));
    }
}
```

Output

```
Enter the size of the array: 5
Enter the elements of the array:
1 2 3 4 5
Maximum product of two elements in the array: 12
```

---

8. Degree of an Array

Problem statement :

Given a non-empty array of non-negative integers nums, the degree of this array is defined as the maximum frequency of any one of its elements.

Your task is to find the smallest possible length of a (contiguous) subarray of nums, that has the same degree as nums

Analysis

- Create a hashmap to store the frequency of each element.
- Create a hashmap to store the first and last occurrence of each element.
- Find the degree of the array.
- Find the length of the subarray with the same degree as the array.

Code

```java
package com.hands_on;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
import java.util.Map;

public class DegreeOfAnArray {
    public static int findShortestSubArray(int[] nums) {
        int max=0;
        HashMap<Integer,Integer> map = new HashMap<>();
        ArrayList<Integer> list = new ArrayList<>();
        for(int i: nums){
            list.add(i);
            map.put(i,map.getOrDefault(i,0)+1);
            if((int)map.get(i)>max)
                max=map.get(i);
        }
        int min = Integer.MAX_VALUE;
        for(Map.Entry m: map.entrySet()){
            if((int)m.getValue()==max){
                int num = (int)m.getKey();
                int n1 = list.indexOf(num);
                int n2 = list.lastIndexOf(num);
                if(n2-n1+1<min)
                    min=n2-n1+1;
            }
        }
        return min;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            nums[i] = sc.nextInt();
        }

        System.out.println("Smallest possible length of a subarray: " + findShortestSubArray(nums));
    }
}
```

Output

```
Enter the size of the array: 7
Enter the elements of the array:
1 2 2 3 1 4 2
Smallest possible length of a subarray: 6
```

---

9. Next Greater Element

Problem statement :

You are given an array 'a' of size 'n'. Print the Next Greater Element(NGE) for every element.

The Next Greater Element for an element 'x' is the first element on the right side of 'x' in the array, which is greater than 'x' If no greater elements exist to the right of 'x',

Analysis

Code

```
package com.hands_on;

import java.util.Scanner;
import java.util.Stack;

public class NextGreaterElement {
    public static void nextGreaterElement(int[] a) {
        Stack<Integer> stack = new Stack<>();
        stack.push(a[0]);
        System.out.print("The next greater element array: ");
        for(int i=1; i<a.length; i++){
            while(!stack.isEmpty() && stack.peek() < a[i]){
                System.out.print(a[i] + " ");
                stack.pop();
            }
            stack.push(a[i]);
        }
        while(!stack.isEmpty()){
            stack.pop();
            System.out.println(-1);
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] a = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            a[i] = sc.nextInt();
        }

        nextGreaterElement(a);
    }
}
```

Output

```
Enter the size of the array: 4
Enter the elements of the array:
7 12 2 20
The next greater element array: 12 20 20 -1
```

10.Single Element in a Sorted Array

Problem statement : You are given a sorted array ‘arr’ of ‘n’ numbers such that every number occurred twice in the array except one, which appears only once.

Return the number that appears once.

Analysis

- The array is sorted.
- The array has all the elements repeated twice except one.
- The single element appears only once.

- Find the single element.
- We can use a hashmap to store the frequency of each element.

Code

```java
package com.hands_on;

import java.util.Scanner;
import java.util.HashMap;

public class SingleElementInASortedArray {
    public static int singleNonDuplicate(int[] nums) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for(int i: nums){
            map.put(i, map.getOrDefault(i, 0)+1);
        }

        for(int i: nums){
            if(map.get(i) == 1){
                return i;
            }
        }

        return -1;
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }

        System.out.println("The single element in the array: " + singleNonDuplicate(arr));
    }
}
```

Output

```
Enter the size of the array: 7
Enter the elements of the array:
1 1 2 2 3 3 4
The single element in the array: 4
```

---

11. Pair Sum

Problem statement

You are given an integer array 'ARR' of size 'N' and an integer 'S'. Your task is to return the list of all pairs of elements such that each sum of elements of each pair equals

> **Note:**
>
> Each pair should be sorted i.e the first value should be less than or equals
>
> to the second value.

Return the list of pairs sorted in non-decreasing order of their first value. In case if two pairs have the same first value, the pair with a smaller second value should com

Analysis

- Sort the array.
- We need to use collections to store the pairs.
- Use two pointers to find the pairs.
- Return the pairs.

Code

```java
package com.hands_on;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class PairSum {
    public static List<List<Integer>> pairSum(int[] arr, int s) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(arr);
        int i=0, j=arr.length-1;
        while(i<j){
            if(arr[i]+arr[j] == s){
                List<Integer> temp = new ArrayList<>();
                temp.add(arr[i]);
                temp.add(arr[j]);
                result.add(temp);
                i++;
                j--;
            }else if(arr[i]+arr[j] < s){
                i++;
            }else{
                j--;
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }
        System.out.print("Enter the sum: ");
        int s = sc.nextInt();

        List<List<Integer>> result = pairSum(arr, s);
        System.out.println("The list of pairs: ");
        for(List<Integer> i: result){
            System.out.println(i);
        }
    }
}
```

Output

```
Enter the size of the array: 5
Enter the elements of the array:
1 2 3 4 5
Enter the sum: 5
The list of pairs:
[1, 4]
[2, 3]
```

---

12. Summary Ranges

Problem statement :

You are given a sorted unique integer array nums.

A range [a,b] is the set of all integers from a to b (inclusive). Return the smallest sorted list of ranges that cover all the numbers in the array exactly. That is, each elemer the ranges, and there is no integer x such that x is in one of the ranges but not in nums. Each range [a,b] in the list should be output as: • "a->b" if a != b • "a" if a == b

Analysis

- We need to use collections to store the ranges.
- Use two pointers to find the ranges.
- Return the ranges.

Code

```java
package com.hands_on;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class SummaryRanges {
    public static List<String> summaryRanges(int[] nums) {
        List<String> result = new ArrayList<>();
        if(nums.length == 0) return result;
        int i=0, j=0;
        while(j<nums.length){
            if(j+1 < nums.length && nums[j+1] == nums[j]+1){
                j++;
            }else{
                if(i == j){
                    result.add(Integer.toString(nums[i]));
                }else{
                    result.add(nums[i] + "->" + nums[j]);
                }
                j++;
                i=j;
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }
```

```
        List<String> result = summaryRanges(arr);
        System.out.println("The list of ranges: ");
        for(String i: result){
            System.out.println(i);
        }
    }
}
```

Output

```
Enter the size of the array: 5
Enter the elements of the array:
0 1 2 4 5
The list of ranges:
0->2
4->5
```

---

13. Kids With the Greatest Number of Candies

Problem statement :

There are n kids with candies. You are given an integer array candies, where each candies[i] represents the number of candies the ith kid has, and an integer extraCand candies that you have. Return a boolean array result of length n, where result[i] is true if, after giving the ith kid all the extraCandies, they will have the greatest numbe otherwise. Note that multiple kids can have the greatest number of candies

Analysis

- We need to use collections to store the candies.
- Find the maximum number of candies.
- Check if the kid has the greatest number of candies after adding the extra candies.
- Return the result.

Code

```
package com.hands_on;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class KidsWithTheGreatestNumberOfCandies {
    public static List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {
        List<Boolean> result = new ArrayList<>();
        int max = Integer.MIN_VALUE;
        for(int i: candies){
            max = Math.max(max, i);
        }
        for(int i: candies){
            if(i+extraCandies >= max){
                result.add(true);
            }else{
                result.add(false);
            }
        }
        return result;
    }

    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }
        System.out.print("Enter the number of extra candies: ");
        int extraCandies = sc.nextInt();

        List<Boolean> result = kidsWithCandies(arr, extraCandies);
        System.out.print("The boolean array: ");
        for(Boolean i: result){
            System.out.print (i+" ");
        }
    }
}
```

Output

```
Enter the size of the array: 5
Enter the elements of the array:
3 2 1 4 5
Enter the number of extra candies: 3
The boolean array: true true false true true
Process finished with exit code 0
```

---

14. Set Mismatch

Problem statement :

You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another r repetition of one number and loss of another number. You are given an integer array nums representing the data status of this set after the error.

Find the number that occurs twice and the number that is missing and return them in the form of an array.

Analysis

- We need to use collections to store the numbers.
- Find the duplicate and the missing number.
- Return the result.

Code

```
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

public class SetMismatch {
    public static int[] findErrorNums(int[] nums) {
        int[] result = new int[2];
        HashMap<Integer, Integer> map = new HashMap<>();
        for(int i: nums){
            map.put(i, map.getOrDefault(i, 0)+1);
        }
        for(int i=1; i<=nums.length; i++){
            if(map.containsKey(i)){
```

```
                if(map.get(i) == 2){
                    result[0] = i;
                }
            }else{
                result[1] = i;
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }

        int[] result = findErrorNums(arr);
        System.out.println("The duplicate and the missing number: ");
        for(int i: result){
            System.out.print(i + " ");
        }
    }
}
```

Output

```
Enter the size of the array: 6
Enter the elements of the array:
1 2 3 3 4 5
The duplicate and the missing number:
3 6
```

---

15. The K Weakest Rows in a Matrix

Problem statement:

You are given an m x n binary matrix mat of 1's (representing soldiers) and 0's (representing civilians). The soldiers are positioned in front of the civilians. That is, all the each row. A row i is weaker than a row j if one of the following is true:

- The number of soldiers in row i is less than the number of soldiers in row j.
- Both rows have the same number of soldiers and i < j.

Return the indices of the k weakest rows in the matrix ordered from weakest to strongest.

Analysis

- We need to use collections to store the rows.
- Find the number of soldiers in each row.
- Sort the rows based on the number of soldiers.
- Return the result.

Code

```
package com.hands_on;
```

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class TheKWeakestRowsInAMatrix {
    public static int[] kWeakestRows(int[][] mat, int k) {
        int[] result = new int[k];
        int[] soldiers = new int[mat.length];
        for(int i = 0; i < mat.length; i++){
            soldiers[i] = Arrays.stream(mat[i]).sum();
        }
        List<Integer> list = new ArrayList<>();
        for(int i = 0; i < mat.length; i++){
            list.add(i);
        }
        list.sort((a, b) -> soldiers[a] - soldiers[b]);
        for(int i = 0; i < k; i++){
            result[i] = list.get(i);
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int m = sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int n = sc.nextInt();
        int[][] arr = new int[m][n];
        System.out.println("Enter the elements of the matrix: ");
        for(int i=0; i<m; i++){
            for(int j=0; j<n; j++){
                arr[i][j] = sc.nextInt();
            }
        }
        System.out.print("Enter the value of k: ");
        int k = sc.nextInt();

        int[] result = kWeakestRows(arr, k);
        System.out.print("The indices of the k weakest rows: ");
        for(int i: result){
            System.out.print(i + " ");
        }
    }
}
```

Output

```
Enter the number of rows: 5
Enter the number of columns: 5
Enter the elements of the matrix:
1 1 0 0 0
1 1 1 1 0
1 0 0 0 0
1 1 1 0 0
1 1 1 1 1
Enter the value of k: 3
The indices of the k weakest rows: 2 0 3
```

---

16. Lucky Numbers in Matrix

Problem statement :

Given an m x n matrix of distinct numbers, return all lucky numbers in the matrix in any order.

A lucky number is an element of the matrix such that it is the minimum element in its row and maximum in its column.

Analysis

- We need to use collections to store the lucky numbers.
- Find the minimum element in each row.
- Find the maximum element in each column.
- Check if the minimum element in the row is the maximum element in the column.
- Return the result.

Code

```
package com.hands_on;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class LuckyNumbersInMatrix {
    public static List<Integer> luckyNumbers (int[][] matrix) {
        List<Integer> result = new ArrayList<>();
        for(int i=0; i<matrix.length; i++){
            int min = Integer.MAX_VALUE;
            int minIndex = 0;
            for(int j=0; j<matrix[i].length; j++){
                if(matrix[i][j] < min){
                    min = matrix[i][j];
                    minIndex = j;
                }
            }
            boolean flag = true;
            for(int j=0; j<matrix.length; j++){
                if(matrix[j][minIndex] > min){
                    flag = false;
                    break;
                }
            }
            if(flag){
                result.add(min);
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int m = sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int n = sc.nextInt();
        int[][] arr = new int[m][n];
        System.out.println("Enter the elements of the matrix: ");
        for(int i=0; i<m; i++){
            for(int j=0; j<n; j++){
                arr[i][j] = sc.nextInt();
            }
        }

        List<Integer> result = luckyNumbers(arr);
```

```
        System.out.print("The lucky numbers in the matrix: ");
        for(int i: result){
            System.out.print(i + " ");
        }
    }
}
```

Output

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter the elements of the matrix:
1 3 2
7 9 8
15 20 21
The lucky numbers in the matrix: 15
```

---

17. Destination City

Problem statement :

You are given the array paths, where paths[i] = [cityAi, cityBi] means there exists a direct path going from cityAi to cityBi. Return the destination city, that is, the city wit It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

Analysis

- We need to use collections to store the cities.
- Find the destination city.
- Return the result.

Code

```
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

public class DestinationCity {
    public static String destCity(String[][] paths) {
        HashMap<String, String> map = new HashMap<>();
        for(String[] path: paths){
            map.put(path[0], path[1]);
        }
        String city = paths[0][0];
        while(map.containsKey(city)){
            city = map.get(city);
        }
        return city;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of paths: ");
        int n = sc.nextInt();
        String[][] arr = new String[n][2];
        System.out.println("Enter the paths: ");
        for(int i = 0; i < n; i++){
            for(int j = 0; j < 2; j++){
                arr[i][j] = sc.next();
            }
```

```
        }

        System.out.println("The destination city: " + destCity(arr));
    }
}
```

Output

```
Enter the number of paths: 4
Enter the paths:
london new-york
new-york lima
lima soa-paule
soa-paule india
The destination city: india
```

---

18. Check if a String Is an Acronym of Words

Problem statement:

Given an array of strings words and a string s, determine if s is an acronym of words.

The string s is considered an acronym of words if it can be formed by concatenating the first character of each string in words in order. For example, "ab" can be formed be formed from ["bear", "aardvark"].

Return true if s is an acronym of words, and false otherwise.

Analysis

- We need to use collections to store the words.
- Find the acronym of the words.
- Return the result.

Code

```
package com.hands_on;

import java.util.Scanner;
import java.util.ArrayList;

public class CheckIfAStringIsAnAcronymOfWords {
    public static boolean isAcronym(String[] words, String s) {
        ArrayList<Character> list = new ArrayList<>();
        for(String word: words){
            list.add(word.charAt(0));
        }
        StringBuilder sb = new StringBuilder();
        for(char c: list){
            sb.append(c);
        }
        return sb.toString().equals(s);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of words: ");
        int n = sc.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the words: ");
```

```
        for(int i = 0; i < n; i++){
            arr[i] = sc.next();
        }
        System.out.print("Enter the string: ");
        String s = sc.next();

        System.out.println(isAcronym(arr, s));
    }
}
```

Output

```
Enter the number of words: 3
Enter the words:
apple mango banana
Enter the string: amb
true
```

---

19. Find Words That Can Be Formed by Characters

Problem statement :

You are given an array of strings words and a string chars.

A string is good if it can be formed by characters from chars (each character can only be used once).

Return the sum of lengths of all good strings in words.

Analysis

- We need to use collections to store the words.
- Find the words that can be formed by the characters.
- Return the sum of the lengths of all good strings.

Code

```
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

public class FindWordsThatCanBeFormedByCharacters {
    public static int countCharacters(String[] words, String chars) {
        int result = 0;
        HashMap<Character, Integer> map = new HashMap<>();
        for(char c: chars.toCharArray()){
            map.put(c, map.getOrDefault(c, 0)+1);
        }
        for(String word: words){
            HashMap<Character, Integer> temp = new HashMap<>(map);
            boolean flag = true;
            for(char c: word.toCharArray()){
                if(temp.containsKey(c) && temp.get(c) > 0){
                    temp.put(c, temp.get(c)-1);
                }else{
                    flag = false;
                    break;
                }
            }
            if(flag){
```

```
                result += word.length();
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of words: ");
        int n = sc.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the words: ");
        for(int i = 0; i < n; i++){
            arr[i] = sc.next();
        }
        System.out.print("Enter the characters: ");
        String chars = sc.next();

        System.out.println("The sum of lengths of all good strings: " + countCharacters(arr, chars));
    }
}
```

Output

```
Enter the number of words: 4
Enter the words:
cat hat tree free
Enter the characters: itachi
The sum of lengths of all good strings: 6
```

---

20. Redistribute Characters to Make All Strings Equal

Problem statement :

You are given an array of strings words (0- indexed).

In one operation, pick two distinct indices i and j, where words[i] is a nonempty string, and move any character from words[i] to any position in words[j].

Return true if you can make every string in words equal using any number of operations, and false otherwise.

Analysis

- We need to use collections to store the words.
- Find the words that can be made equal.
- Return the result.

Code

```
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

public class RedistributeCharactersToMakeAllStringsEqual {
    public static boolean makeEqual(String[] words) {
        HashMap<Character, Integer> map = new HashMap<>();
        for(String word: words){
            for(char c: word.toCharArray()){
                map.put(c, map.getOrDefault(c, 0)+1);
            }
```

```
        }
        for(int i: map.values()){
            if(i % words.length != 0){
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of words: ");
        int n = sc.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the words: ");
        for(int i = 0; i < n; i++){
            arr[i] = sc.next();
        }

        System.out.println(makeEqual(arr));
    }
}
```

Output

```
Enter the number of words: 3
Enter the words:
abc aabc bc
true
```

---

21. First Unique Character in a String

Problem statement :

Given a string s, find the first non-repeating character in it and return its index. If it does not exist, return -1.

Analysis

- We need to use collections to store the characters.
- Find the first non-repeating character in the string.
- Return the index of the first non-repeating character.

Code

```
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

public class FirstUniqueCharacterInAString {
    public static int firstUniqChar(String s) {
        HashMap<Character, Integer> map = new HashMap<>();
        for(char c: s.toCharArray()){
            map.put(c, map.getOrDefault(c, 0)+1);
        }
        for(int i=0; i<s.length(); i++){
            if(map.get(s.charAt(i)) == 1){
                return i;
            }
```

```
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the string: ");
        String s = sc.next();

        System.out.println("The index of the first non-repeating character: " + firstUniqChar(s));
    }
}
```

Output

```
Enter the string: shabari
The index of the first non-repeating character: 0
```

---

22. Ransom Note

Problem statement :

Given two strings ransomNote and magazine, return true if ransomNote can be constructed by using the letters from magazine and false otherwise.

Each letter in magazine can only be used once in ransomNote.

Analysis

- We need to use collections to store the characters.
- Find the ransom note can be constructed by using the letters from the magazine.
- Return the result.

Code

```
package com.hands_on;

import java.util.HashMap;
import java.util.Scanner;

public class RansomNote {
    public static boolean canConstruct(String ransomNote, String magazine) {
        HashMap<Character, Integer> map = new HashMap<>();
        for(char c: magazine.toCharArray()){
            map.put(c, map.getOrDefault(c, 0)+1);
        }
        for(char c: ransomNote.toCharArray()){
            if(map.containsKey(c) && map.get(c) > 0){
                map.put(c, map.get(c)-1);
            }else{
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the ransom note: ");
        String ransomNote = sc.next();
        System.out.print("Enter the magazine: ");
```

```
        String magazine = sc.next();

        System.out.println(canConstruct(ransomNote, magazine));
    }
}
```

Output

```
Enter the ransom note: shabari
Enter the magazine: shabari K S
true
```

---

23. Unique Morse Code Words

Problem statement :

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:

- 'a' maps to ".-",
- 'b' maps to "-...",
- 'c' maps to "-.-.", and so on.

For convenience, the full table for the 26 letters of the English alphabet is given below:

```
[
".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",
".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",
".--","-..-","-.--","--.."
]
```

Given an array of strings words where each word can be written as a concatenation of the Morse code of each letter.

- For example, "cab" can be written as "-.-..--...", which is the concatenation of "-.-.", ".-", and "-...". We will call such a concatenation the transformation of a word.

Return the number of different transformations among all words we have

Analysis

- We need to use collections to store the Morse code.
- Find the number of different transformations among all words.
- Return the result.

Code

```
package com.hands_on;

import java.util.HashSet;
import java.util.Scanner;

public class UniqueMorseCodeWords {
    public static int uniqueMorseRepresentations(String[] words) {
        String[] morse = {".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",
                ".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",
                ".--","-..-","-.--","--.."};
        HashSet<String> set = new HashSet<>();
        for(String word: words){
            StringBuilder sb = new StringBuilder();
            for(char c: word.toCharArray()){
```

```
                sb.append(morse[c-'a']);
            }
            set.add(sb.toString());
        }
        return set.size();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of words: ");
        int n = sc.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the words: ");
        for(int i = 0; i < n; i++){
            arr[i] = sc.next();
        }

        System.out.println("The number of different transformations: " + uniqueMorseRepresentations(arr));
    }
}
```

Output

```
Enter the number of words: 4
Enter the words:
gin gig zin msg
The number of different transformations: 3
```

---

24. Valid Parentheses

Problem statement : You're given a string 'S' consisting of "{", "}", "(", ")", "[" and "]" .

Return true if the given string 'S' is balanced, else return false.

For example:

'S' = "{}()".

There is always an opening brace before a closing brace i.e. '{' before '}', '(' before ').

So the 'S' is Balanced.

Analysis

- We need to use collections to store the parentheses.
- Find the valid parentheses.
- Return the result.

Code

```
package com.hands_on;

import java.util.Scanner;
import java.util.Stack;

public class ValidParentheses {
    public static boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for(char c: s.toCharArray()){
            if(c == '(' || c == '{' || c == '['){
```

```
                stack.push(c);
            }else{
                if(stack.isEmpty()){
                    return false;
                }
                char top = stack.pop();
                if((c == ')' && top != '(') || (c == '}' && top != '{') || (c == ']' && top != '[')){
                    return false;
                }
            }
        }
        return stack.isEmpty();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the string: ");
        String s = sc.next();

        System.out.println(isValid(s));
    }
}
```

Output

```
Enter the string: {}()[({})]
true
```

---

25. Reverse Linked List

Problem statement :

Given the head of a singly linked list, reverse the list, and return the reversed list.

Analysis

- We need to use collections to store the linked list.
- Find the reverse of the linked list.
- Print the result.

Code:

```
package com.hands_on;

import java.util.LinkedList;
import java.util.Scanner;

public class ReverseLinkedList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the elements of the Linked List: ");
        LinkedList<Integer> linkedList = new LinkedList<>();
        while (true){
            int a = sc.nextInt();
            if (a == -1)
                break;
            linkedList.add(a);
        }
        System.out.println("Reverse of the Linked List: "+ linkedList.reversed());
```

```
    }
}
```

Output

```
Enter the elements of the Linked List: 1 2 3 1 2 3 -1
Reverse of the Linked List: [3, 2, 1, 3, 2, 1]
```

---

26. Palindrome Linked List

Problem statement :

You are given a singly Linked List of integers. Your task is to return true if the given singly linked list is a palindrome otherwise returns false.

For example:

The given linked list is 1 -> 2 -> 3 -> 2-> 1-> NULL.

It is a palindrome linked list because the given linked list has the same order of elements when traversed forwards and backward.

Analysis

- We need to use collections to store the linked list.
- Find the palindrome linked list.
- Print the result.

Code:

```java
package com.hands_on;

import java.util.LinkedList;
import java.util.Scanner;

public class PalindromeLinkedList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the elements of the Linked List: ");
        LinkedList<Integer> linkedList = new LinkedList<>();
        while (true){
            int a = sc.nextInt();
            if (a == -1)
                break;
            linkedList.add(a);
        }
        LinkedList<Integer> linkedListOriginal = linkedList;
        System.out.println("Is the Linked List is Palindrome? "+ linkedListOriginal.equals(linkedList.reversed()));
    }
}
```

Output

```
Enter the elements of the Linked List: 1 2 3 2 1 -1
Is the Linked List is Palindrome? true
```

---

27. Reverse First K elements of Queue

Problem statement :

You are given a QUEUE containing `N` integers and an integer `K`. You need to reverse the order of the first `K` elements of the queue, leaving the other elements in the s
standard operations of the QUEUE STL:

1. enqueue(x) : Adds an item x to rear of the queue
2. dequeue() : Removes an item from front of the queue
3. size() : Returns number of elements in the queue.
4. front() : Finds the front element.

Analysis

- We need to use collections to store the queue.
- Find the reverse of the first `K` elements of the queue.
- Print the result.

Code

```
package com.hands_on;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;
import java.util.Stack;

public class ReverseFirstKElementsOfQueue {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the Queue: ");
        int n = sc.nextInt();
        Queue<Integer> queue = new LinkedList<>();
        System.out.println("Enter the elements of the Queue: ");
        for(int i=0; i<n; i++){
            queue.add(sc.nextInt());
        }
        System.out.print("Enter the value of K: ");
        int k = sc.nextInt();
        Stack<Integer> stack = new Stack<>();
        for(int i=0; i<k; i++){
            stack.push(queue.poll());
        }
        while(!stack.isEmpty()){
            queue.add(stack.pop());
        }
        for(int i=0; i<n-k; i++){
            queue.add(queue.poll());
        }
        System.out.println("The Queue after reversing the first K elements: " + queue);
    }
}
```

Output

```
Enter the number of elements in the Queue: 5
Enter the elements of the Queue:
1 2 3 4 5
Enter the value of K: 3
The Queue after reversing the first K elements: [3, 2, 1, 4, 5]
```

28. Delete middle element from stack

Problem statement :

You are having a stack "ARR" of size 'N+1', your task is to delete the middlemost element so that the size of resulting stack is 'N'. A stack is a linear data structure where take place at the top. It follows FILO (First In Last Out) or LIFO (Last In First Out) approaches. Books piled on top of each other is an example of a stack, where you can o which is at the top of the stack. Likewise, you can only add a single book at a time, on the top of the stack only

Analysis

- We need to use collections to store the stack.
- Find the middlemost element of the stack.
- Delete the middlemost element of the stack.
- Print the result.
- We need to use recursion to delete the middlemost element of the stack.

Code

```
package com.hands_on;

import java.util.Scanner;
import java.util.Stack;

public class DeleteMiddleElementFromStack {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the Stack: ");
        int n = sc.nextInt();
        Stack<Integer> stack = new Stack<>();
        System.out.println("Enter the elements of the Stack: ");
        for(int i=0; i<n; i++){
            stack.push(sc.nextInt());
        }
        Stack<Integer> temp = new Stack<>();
        int mid = (n+1)/2;
        for(int i=0; i<mid-1; i++){
            temp.push(stack.pop());
        }
        stack.pop();
        while(!temp.isEmpty()){
            stack.push(temp.pop());
        }
        System.out.println("The Stack after deleting the middlemost element: " + stack);
    }
}
```

Output

```
Enter the number of elements in the Stack: 7
Enter the elements of the Stack:
1 2 3 4 3 2 1
The Stack after deleting the middlemost element: [1, 2, 3, 3, 2, 1]
```

29. Flatten A Linked List

Problem statement :

You are given a linked list containing 'n' 'head' nodes, where every node in the linked list contains two pointers:

1. ‘next’ which points to the next node in the list
2. ‘child’ pointer to a linked list where the current node is the head.

Each of these child linked lists is in sorted order and connected by 'child' pointer.

Your task is to flatten this linked such that all nodes appear in a single layer or level in a 'sorted order'

Analysis

- Use ArrayList to get the input from the user
- Traverse the ArrayList and add the elements to the linked list in sorted order
- Print the linked list

Code:

```
package com.hands_on;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Scanner;

public class FlattenALinkedList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of head nodes: ");
        int n = sc.nextInt();
        LinkedList<Integer> linkedList = new LinkedList<>();
        ArrayList<LinkedList<Integer>> list = new ArrayList<>();
        for(int i=0; i<n; i++){
            System.out.print("Enter the number of elements in the child linked list: ");
            int m = sc.nextInt();
            LinkedList<Integer> childList = new LinkedList<>();
            for(int j=0; j<m; j++){
                childList.add(sc.nextInt());
            }
            list.add(childList);
        }
        for(LinkedList<Integer> l: list){
            for(int i: l){
                linkedList.add(i);
            }
        }
        linkedList.sort(null);
        System.out.println("The Flattened Linked List: " + linkedList);
    }
}
```

Output

```
Enter the number of head nodes: 5
Enter the number of elements in the child linked list: 3
1 2 3
Enter the number of elements in the child linked list: 3
4 5 6
Enter the number of elements in the child linked list: 3
7 8 9
Enter the number of elements in the child linked list: 3
10 11 12
```

```
Enter the number of elements in the child linked list: 3
13 14 15
The Flattened Linked List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

---

30. Middle Of Linked List

Problem statement :

Given a singly linked list of 'N' nodes. The objective is to determine the middle node of a singly linked list. However, if the list has an even number of nodes, we return t

Analysis

- We need to use collections to store the linked list.
- Find the middle node of the linked list.
- Print the result.

Code

```java
package com.hands_on;

import java.util.LinkedList;
import java.util.Scanner;

public class MiddleOfLinkedList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the Linked List: ");
        int n = sc.nextInt();
        LinkedList<Integer> linkedList = new LinkedList<>();
        System.out.println("Enter the elements of the Linked List: ");
        for (int i = 0; i < n; i++){
            linkedList.add(sc.nextInt());
        }
        int mid = (n)/2;
        System.out.print("The Linked List after the middle node: ");
        for ( int i = mid; i < n; i++) {
            System.out.print(linkedList.get(i) + " ");
        }
    }
}
```

Output

```
Enter the number of elements in the Linked List: 6
Enter the elements of the Linked List:
1 2 3 4 5 6
The Linked List after the middle node: 4 5 6
```