Daily Challenge **Happy Coding from necse** **SkillRack**

Matrix Layers - Repeated Sum

The program must accept an integer matrix of size N*N as the input. The program must find the sum of integers in each layer of the matrix (from outer layer to inner layer). Then the program must print the repeated sum among the obtained sum values. If two or more sum values are repeated, then the program must print the first occurring sum as the output. If there is no repeated sum, then the program must print -1 as the output.

Boundary Condition(s):
3 <= N <= 50
0 <= Matrix element value <= 1000

Input Format:
The first line contains N.
The next N lines each contain N integers separated by a space.

Output Format:
The first line contains an integer representing the repeated sum or -1 based on the given conditions.

Example Input/Output 1:
Input:
5
2 3 4 5 1
3 10 6 7 2
4 13 6 9 2
5 6 5 3 3
7 5 7 2 4

Output:
59

Explanation:
1st layer sum: 59
2nd layer sum: 59
3rd layer sum: 6
Here the sum 59 is repeated, so 59 is printed as the output.

Example Input/Output 2:
Input:
7
76 56 96 69 12 75 81
76 69 64 95 57 62 53
84 32 64 24 18 51 10
99 8 64 22 12 88 23
52 94 28 93 43 95 97
40 1 91 8 56 84 50
76 65 20 87 97 45 80

Output:
-1

Example Input/Output 3:
Input:
8
1 1 1 1 1 1 1 1
1 2 2 2 2 2 2 1
1 1 1 1 1 1 2 1
1 1 1 3 3 1 2 1
1 1 1 3 3 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1

Output:
28

Max Execution Time Limit: 50 millisecs

Ambiance

Java (12.0)

```java
import java.util.*;
public class Hello {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        char dir = 'r';

        int row=0,col=0;

        int n = sc.nextInt();
        int arr[][]= new int[n][n];

        Map<Integer,Integer> ht = new LinkedHashMap<>();

        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                arr[i][j]=sc.nextInt();

        int temp_sum=0,maxx=0;
        boolean close=false;

        for(int i=0;i<n;i++){
            if(close) break;
            temp_sum=0;

            //System.out.println("current-boundry-"+n+"-"+i);

            while(true){
                //System.out.println(row+"-"+col);

                temp_sum+=arr[row][col];

                if(dir=='r'){
                    if(col+1>=n-i){
                        row++;
                        dir='d';
                    }else col++;

                }else if(dir=='d'){
                    if(row+1>=n-i){
                        col--;
                        dir='l';
                    }else row++;

                }else if(dir=='l'){
                    if(col-1<=-1+i){
                        row--;
                        dir='u';
                    }else col--;

                }else if(dir=='u'){
                    if(row-1 <= +i){
                        row=i+1;col=i+1;dir='r';
                        if(row==n/2){
                            close=true;
                            if(ht.containsKey(temp_sum)){
                                ht.put(temp_sum,ht.get(temp_sum)+1);
                            }
                        }
                        break;
                    }
                    else row--;
                }

            }
            if(ht.containsKey(temp_sum)){
                ht.put(temp_sum,ht.get(temp_sum)+1);
                maxx = Math.max(maxx,ht.get(temp_sum));
            }else{
                ht.put(temp_sum,1);
            }
        }

        if(maxx==0){
            System.out.println(-1);
            return;
```

```
        }

        for(Map.Entry<Integer,Integer> m:ht.entrySet()){
            if(m.getValue()==maxx){
                System.out.println(m.getKey());
            }
        }



    }
}
```

1912067@nec

Please wait while we run the program ....