TCS NQT 2025

15th march shift 1

Ques 5:

Problem Statement: Making Triplets Equal

You are given a triplet of integers (a,b,c)(a, b, c). You can perform the following operation any number of times:

1. **Select any two numbers from the triplet.**
2. **Add 1 to both selected numbers.**
3. **Subtract 1 from the remaining number.**

Your task is to determine whether it is possible to make all three numbers equal using the given operations.

Input:

- **A single integer TT representing the number of test cases.**
- **Each test case consists of three integers a,b,ca, b, c.**

Output:

- **For each test case, print "YES" if it is possible to make all three numbers equal; otherwise, print "NO".**

Constraints:

$$1 \leq T \leq 10^4$$

$$-10^9 \leq a, b, c \leq 10^9$$

Example:

Input:

3
1 2 3
4 4 4
3 3 7

Output:

-1
0
2

C++

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int min_steps_to_equal(int P, int Q, int R) {
    int arr[3] = {P, Q, R};
    sort(arr, arr + 3);

    if (arr[0] == arr[1] && arr[1] == arr[2])
        return 0;

    int steps = 0;
    while (true) {
        arr[0] += 1;
        arr[1] += 1;
        arr[2] -= 1;
        steps++;
        sort(arr, arr + 3);

        if (arr[0] == arr[1] && arr[1] == arr[2])
            return steps;

        if ((arr[0] == arr[1] && arr[1] + 1 == arr[2]) ||
            (arr[1] == arr[2] && arr[0] + 1 == arr[1])) {
            return -1;
        }
    }
}

int main() {
    int T;
    cin >> T;  // Number of test cases
    while (T--) {
        int P, Q, R;
        cin >> P >> Q >> R;
        cout << min_steps_to_equal(P, Q, R) << endl;
    }
    return 0;
}
```

PYTHON

```
def min_steps_to_equal(P, Q, R):
    arr = [P, Q, R]
    arr.sort()

    if arr[0] == arr[1] == arr[2]:
        return 0

    steps = 0
    while True:
        arr[0] += 1
        arr[1] += 1
        arr[2] -= 1
        steps += 1
        arr.sort()

        if arr[0] == arr[1] == arr[2]:
            return steps

        if (arr[0] == arr[1] and arr[1] + 1 == arr[2]) or \
           (arr[1] == arr[2] and arr[0] + 1 == arr[1]):
            return -1

# Input handling
T = int(input())  # Number of test cases
for _ in range(T):
    P, Q, R = map(int, input().split())
    print(min_steps_to_equal(P, Q, R))
```

JAVA

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

public class Main {
  public static int minStepsToEqual(int P, int Q, int R) {
    int[] arr = {P, Q, R};
    Arrays.sort(arr);

    if (arr[0] == arr[1] && arr[1] == arr[2]) {
      return 0;
    }

    int steps = 0;
    while (true) {
      arr[0] += 1;
      arr[1] += 1;
      arr[2] -= 1;
      steps++;
      Arrays.sort(arr);

      if (arr[0] == arr[1] && arr[1] == arr[2]) {
        return steps;
      }

      if ((arr[0] == arr[1] && arr[1] + 1 == arr[2]) ||
        (arr[1] == arr[2] && arr[0] + 1 == arr[1])) {
        return -1;
      }
    }
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int T = sc.nextInt(); // Number of test cases

    while (T-- > 0) {
      int P = sc.nextInt();
      int Q = sc.nextInt();
      int R = sc.nextInt();
      System.out.println(minStepsToEqual(P, Q, R));
    }
    sc.close();
  }
}
```

Ques 2:

Problem Statement: Range Sum Query

You are given two integers i and j, where 0≤i<j≤9999. Your task is to compute the sum of all integers from index i to j, inclusive.

Input Format:

- **A single integer T representing the number of queries.**
- **Each of the next T lines contains two integers i and j (0≤i<j≤9999)**

Output Format:

- **For each query, print a single integer representing the sum of numbers from i to j.**

Constraints: $1 \leq T \leq 10^4$

$0 \leq i < j \leq 9999$

Example Input:

3
0 3
2 6
10 1001

Example Output:

6 20 invalid input i&j i <=j<10000

Explanation:

1. **Sum from 0 to 3: 0+1+2+3=6**
2. **Sum from 2 to 6: 2+3+4+5+6=20**
3. **Sum from 10 to 20: 10+11+...+20=165**

MASSIVE SUCCESS RATE

"Transform Your Interview Opportunity into an Offer Letter and Make Your Parents Proud!"

- In-depth Technical Mock
  - Crack coding challenges with real experts.
- HR & Managerial Prep
  - Master behavioral questions and impress TCS Interviewer.
- Full Interview Simulation
  - Ace both technical and HR in one session.
- Resume Review
  - Identify and fix weaknesses for a standout CV.
- Personalized Feedback & Expert Guidance
  - Tailored improvement tips to boost success.

www.primecoding.in

JAVA

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

class MyClass{
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    ArrayList<Integer> arr = new ArrayList<>();
    String str = sc.nextLine();
    boolean flag = false;
    StringBuilder sb = new StringBuilder();
    for(int i=0; i<str.length(); i++) {
      if(str.charAt(i) == '-') {
        System.out.print("Invalid, out of range");
        flag = true;
        break;
      }
      if(Character.isDigit(str.charAt(i))) {
        sb.append(str.charAt(i));
      } else if(sb.length() > 0) {
        int temp = Integer.parseInt(sb.toString());
        if(temp < 0 || temp > 10000) {
          System.out.print("Invalid, out of range");
          flag = true;
          break;
        }
        arr.add(temp);
        sb.setLength(0);
      }
    }
    if(sb.length()>0) {
      arr.add(Integer.parseInt(sb.toString()));
    }

    if(!flag) {
      if(arr.size() < 2) {
        System.out.print("Invalid more than one input needed");
      } else {
        for(int i=0; i<arr.size(); i+=2) {
          int ans = 0;
          for(int j=arr.get(i); j<arr.get(i+1); j++) {
            ans+=j;
          }
          System.out.print(ans + " ");
        }
      }
    }
  }
}
```

```java
import java.util.Scanner;

public class Main {
  public static int rangeSum(int i, int j) {
    return (j * (j + 1) / 2) - (i * (i - 1) / 2);
  }

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

    for (int t = 0; t < T; t++) {
      String userInput = sc.nextLine().trim();
      String[] values = userInput.split(" ");

      if (values.length < 2) {
        System.out.println("Invalid input i&j i <= j < 10000");
        continue;
      }

      try {
        int i = Integer.parseInt(values[0]);
        int j = Integer.parseInt(values[1]);

        if (i >= j || i < 0 || j >= 10000) {
          System.out.println("Invalid input i&j i <= j < 10000");
        } else {
          System.out.print(rangeSum(i, j) + " ");
        }
      } catch (NumberFormatException e) {
        System.out.println("Invalid input i&j i <= j < 10000");
      }
    }

    sc.close();
  }
}
```

```cpp
#include <iostream>
#include <sstream>
using namespace std;

int rangeSum(int i, int j) {
  return (j * (j + 1) / 2) - (i * (i - 1) / 2);
}

int main() {
  int T;
  cin >> T;
  cin.ignore();

  for (int t = 0; t < T; ++t) {
    string userInput;
    getline(cin, userInput);
    stringstream ss(userInput);

    int i, j;
    if (!(ss >> i >> j)) {
      cout << "Invalid input i&j i <= j < 10000" << endl;
      continue;
    }

    if (i >= j || i < 0 || j >= 10000) {
      cout << "Invalid input i&j i <= j < 10000" << endl;
    } else {
      cout << rangeSum(i, j) << " ";
    }
  }

  return 0;
}
```

PYTHON

```
def range_sum(i, j):
    return (j * (j + 1) // 2) - (i * (i - 1) // 2)

T = int(input())

for _ in range(T):
    user_input = input()
    values = user_input.split()
    if len(values) < 2:
        print("Invalid input i&j i  <= j < 10000")
        continue

    i, j = map(int, values)

    if i >= j or i < 0 or j >= 10000:
        print("invalid input i&j i  <= j < 10000")
    else:
        print(range_sum(i, j), end = " ")
```