1，沫璃有一个画板，画板可以抽象成有 100 行每行 100 个像素点的正方形。沫璃在画板上画画，她一共画了 n 次，每次将一个矩形涂上颜色。沫璃想知道一共有多少个像素点被她涂过颜色。若一个像素点被涂了 k 次，那么认为有 k 个像素点被涂过颜色

```
/* 输入例子 1:
*2
*2
*1 1 2 3
*2 2 3 3
*2
*1 1 3 3
*1 1 3 3
*
* 从（1,1）点填涂到（2,3）点，一共涂了 6 个格子（包括了点（1,1），一开始没考虑这个点就懵了，直接跳下一题），
* 用下图解释下这一操作（0 表示已经涂色，1 表示未涂色）
*0 0 0 1 1 1 1...
*0 0 0 1 1 1 1...
*1 1 1 1 1 1 1...
* . .
* . .
* 同理从点（2,2）涂到（3,3），一共涂了 4 个格子，所以用例输出 10；第二个用例输出 18.
*/
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    for (int i = 0; i < n; i++) {
      int t = sc.nextInt();
      int sum=0;
      for (int j = 0; j < t; j++) {
        int x1 = sc.nextInt();
        int y1 = sc.nextInt();
        int x2 = sc.nextInt();
        int y2 = sc.nextInt();
        sum+=(x2-x1+1)*(y2-y1+1);
      }
      System.out.println(sum);
    }
  }
}
```

2，沫璃发起了一场交易，她将她的 5 个朋友聚在一起准备进行一场交易。交易开始前，大家各有 b(b>0)个硬币，交易后，每个人有 ai 个硬币。由于硬币不方面携带，在交易过程中可能会丢失。现在沫璃想知道是否一定丢失硬币，或者在可能没有丢失硬币的情况下，交易

前每个人的硬币数 b。沫璃只是组织者，不参与交易。

```
import java.util.Scanner;
//没啥好说的....能被 5 整除就是 YES，否则为 NO，特殊情况是： 都为 0 的时候是 NO
/**
* 沫璃发起了一场交易,她将她的 5 个朋友聚在一起准备进行一场交易。交易开始前,大家各有 b(b>0)个硬币,
* 交易后，每个人有 ai 个硬币。由于硬币不方面携带，在交易过程中可能会丢失。
* 现在沫璃想知道是否一定丢失硬币，或者在可能没有丢失硬币的情况下，交易前每个人的硬币数 b。沫璃只是组织者，不参与交易。
*/
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    for (int i = 0; i < n; i++) {
      int sum =0;
      for (int j = 0; j < 5; j++) {
        int num = sc.nextInt();
        sum+=num;
      }
      if(sum%5==0 && sum!=0){
        System.out.println(sum/5);
      }else {
        System.out.println("-1");
      }
    }
  }
}
```

3 沫璃邀请她的朋友参加周末的派对。沫璃买了 3 种颜色的气球，现在她要有这些气球来装饰餐桌，每个餐桌只用恰好 3 个气球装饰，要求 3 个气球的颜色不能完全一样，可以是 2 种或者 3 种颜色。沫璃想知道这些气球最多能装饰多少张餐桌。

```java
import java.util.Arrays;
import java.util.Scanner;
/**
* 沫璃邀请她的朋友参加周末的派对。沫璃买了 3 种颜色的气球，现在她要有这些气球来装饰餐桌，
* 每个餐桌只用恰好 3 个气球装饰，要求 3 个气球的颜色不能完全一样，可以是 2 种或者 3 种颜色。
* 沫璃想知道这些气球最多能装饰多少张餐桌。
*/
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    for (int i = 0; i < n; i++) {
      long[] num = new long[3];
      for (int j = 0; j < 3; j++) {
        num[j] = sc.nextLong();
      }
      Arrays.sort(num);
      if(num[2]/2>=num[0]+num[1]){
        System.out.println(num[0]+num[1]);
      }else
        System.out.println((num[0]+num[1]+num[2])/3);
    }
  }
}
```

3，茉莉有 2n 匹马，每匹马都有一个速度 v，现在茉莉将马分为两个队伍，每个队伍各有 n 匹马，两个队之间进行 n 场比赛，每场比赛两队各派出一匹马参赛，每匹马都恰好出场一次。茉莉想知道是否存在一种分配队伍的方法使得无论怎么安排比赛，第一个队伍都一定能获的全胜，两匹马若速度不同，那么速度快的获胜，若速度一样，则都有可能获胜。

```java
import java.util.Arrays;
import java.util.Scanner;
//排下序，第一组最差的马比第二组最强的马强就是 YES，否则 NO
/**
* ***有 2n 匹马，每匹马都有一个速度 v，现在***将马分为两个队伍，每个队伍各有 n 匹马，
* 两个队之间进行 n 场比赛，每场比赛两队各派出一匹马参赛，每匹马都恰好出场一次。
* ***想知道是否存在一种分配队伍的方法使得无论怎么安排比赛，第一个队伍都一定能获的全胜，
* 两匹马若速度一样，那么速度快的获胜，若速度一样，则都有可能获胜。
*/
public class Main {
  public static void main(String[] args) {
    Scanner sc= new Scanner(System.in);
    int n = sc.nextInt();
    for (int i = 0; i < n; i++) {
      int t = sc.nextInt();
      int[] nums = new int[t*2];
```

```
        for (int j = 0; j < t*2; j++) {
            nums[j] = sc.nextInt();
        }
        Arrays.sort(nums);
        if(nums[t]>nums[t-1]){
            System.out.println("YES");
        }else {
            System.out.println("NO");
        }
    }
  }
}
```

4，有 K 种不同的玫瑰花，现在要摆放在 N 个位置上，要求每种颜色的花至少出现过一次，请问有多少种不同的方案数呢？，因为答案可能很大，你只需要输出它对 772235 取余后的结果.

链接：

https://www.nowcoder.com/questionTerminal/7a1df018c11e46d7873efe994e9b21e1

来源：牛客网

```
import java.util.Scanner;

public class rose {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int N=sc.nextInt();
        int M=sc.nextInt();
        long a=jiecheng(N);
        long b=jiecheng(M-N);
        long c=a/b;
        System.out.println(c);

    }
    public static long jiecheng(int a){
        long b=1;
        for (int i=1;i<=a;i++){
            b=b*i;
        }
        return b;


    }
}
```