1. 三个同样的字母连在一起，一定是拼写错误，去掉一个的就好啦：比如 helllo -> hello
2. 两对一样的字母（AABB 型）连在一起，一定是拼写错误，去掉第二对的一个字母就好啦：比如 helloo -> hello
3. 上面的规则优先“从左到右”匹配，即如果是 AABBCC，虽然 AABB 和 BBCC 都是错误拼写，应该优先考虑修复 AABB，结果为 AABCC
请听题：请实现大锤的自动校对程序

```
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int line = scanner.nextInt();
      scanner.nextLine();
      for (int i = 0; i < line; i++) {
          System.out.println(scanner.nextLine().replaceAll("(.)\\1+","$1$1").replaceAll("(.)\\1(.)\\2",
"$1$1$2"));
      }
   }
}
```

我叫王大锤，是一名特工。我刚刚接到任务：在字节跳动大街进行埋伏，抓捕恐怖分子孔连顺。和我一起行动的还有另外两名特工，我提议

1. 我们在字节跳动大街的 N 个建筑中选定 3 个埋伏地点。
2. 为了相互照应，我们决定相距最远的两名特工间的距离不超过 D。

我特喵是个天才！经过精密的计算，我们从 X 种可行的埋伏方案中选择了一种。这个方案万无一失，颤抖吧，孔连顺！
……
万万没想到，计划还是失败了，孔连顺化妆成小龙女，混在 cosplay 的队伍中逃出了字节跳动大街。只怪他的伪装太成功了，就是杨过本人来了也发现不了的！

请听题：给定 N（可选作为埋伏点的建筑物数）、D（相距最远的两名特工间的距离的最大值）以及可选建筑的坐标，计算在这次行动中，大锤的小队有多少种埋伏选择。
注意：
1. 两个特工不能埋伏在同一地点
2. 三个特工是等价的：即同样的位置组合(A, B, C) 只算一种埋伏方法，不能因“特工之间互换位置”而重复使用

```
import java.util.*;

public class Main {
   private int mod = 99997867;
```

```
    private void sln() {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt(), D = sc.nextInt();
        long cnt = 0;
        if (N <= 2) {
            System.out.println(-1);
            return;
        }
        int[] locs = new int[N];
        for (int i = 0; i < N; i++) {
            locs[i] = sc.nextInt();
        }
        sc.close();
        int left = 0, right = 2;
        while (right < N) {
            if (locs[right] - locs[left] > D) left++;
            else if (right - left < 2) right++;
            else {
                cnt += calC(right - left);
                right++;
            }
        }
        cnt %= mod;
        System.out.println(cnt);
    }

    private long calC(long num) {
        return num * (num - 1) / 2;
    }


    public static void main(String[] args) {
        new Main().sln();
    }
}
```

小包最近迷上了一款叫做雀魂的麻将游戏，但是这个游戏规则太复杂，小包玩了几个月了还是输多赢少。
于是生气的小包根据游戏简化了一下规则发明了一种新的麻将，只留下一种花色，并且去除了一些特殊和牌方式（例如七对子等），具体的规则如下：

总共有 36 张牌，每张牌是 1~9。每个数字 4 张牌。
你手里有其中的 14 张牌，如果这 14 张牌满足如下条件，即算作和牌

14 张牌中有 2 张相同数字的牌，称为雀头。
除去上述 2 张牌，剩下 12 张牌可以组成 4 个顺子或刻子。顺子的意思是递增的连续 3 个数字牌（例如 234,567 等），刻子的意思是相同数字的 3 个数字牌（例如 111,777）

例如：
1 1 1 2 2 2 6 6 6 7 7 7 9 9 可以组成 1,2,6,7 的 4 个刻子和 9 的雀头，可以和牌
1 1 1 1 2 2 3 3 5 6 7 7 8 9 用 1 做雀头，组 123,123,567,789 的四个顺子，可以和牌
1 1 1 2 2 2 3 3 3 5 6 7 7 9 无论用 1 2 3 7 哪个做雀头，都无法组成和牌的条件。

现在，小包从 36 张牌中抽取了 13 张牌，他想知道在剩下的 23 张牌中，再取一张牌，取到哪几种数字牌可以和牌。

```
import java.util.*;

public class Main {

  private void sln() {
    Scanner sc = new Scanner(System.in);
    int[] state = new int[9], helpArr = new int[9];
    ArrayList<Integer> res = new ArrayList<>();
    for (int i = 0; i < 13; i++) {
      int num = sc.nextInt();
      state[num - 1]++;
    }
    for (int i = 0; i < 9; i++) {
      if (state[i] < 4) {
        int num = i + 1;
        System.arraycopy(state, 0, helpArr, 0, 9);
        helpArr[i]++;
        if (canHu(helpArr, 14, false)) res.add(num);
      }
    }
    if (res.isEmpty()) System.out.println(0);
    else {
      StringBuffer sbf = new StringBuffer();
      sbf.append(res.get(0));
      for (int i = 1; i < res.size(); i++) {
        sbf.append(" ");
        sbf.append(res.get(i));
      }
      System.out.println(sbf.toString());
    }
  }

  private boolean canHu(int[] arr, int total, boolean hasHead) {
```

```
        if (total == 0) return true;
        if (!hasHead) {
            for (int i = 0; i < 9; i++) {
                if (arr[i] >= 2) {
                    arr[i] -= 2;
                    if (canHu(arr, total - 2, true)) return true;
                    arr[i] += 2;
                }
            }
            return false;
        } else {
            for (int i = 0; i < 9; i++) {
                if (arr[i] > 0) {
                    if (arr[i] >= 3) {
                        arr[i] -= 3;
                        if (canHu(arr, total - 3, true)) return true;
                        arr[i] += 3;
                    }
                    if (i + 2 < 9 && arr[i + 1] > 0 && arr[i + 2] > 0) {
                        arr[i]--;
                        arr[i + 1]--;
                        arr[i + 2]--;
                        if (canHu(arr, total - 3, true)) return true;
                        arr[i]++;
                        arr[i + 1]++;
                        arr[i + 2]++;
                    }
                }
            }
        }
        return false;
    }

    public static void main(String[] args) {
        new Main().sln();
    }
}
```

小明是一名算法工程师，同时也是一名铲屎官。某天，他突发奇想，想从猫咪的视频里挖掘一些猫咪的运动信息。为了提取运动信息，他需要从视频的每一帧提取“猫咪特征”。一个猫咪特征是一个两维的 vector<x, y>。如果 x_1=x_2 and y_1=y_2，那么这俩是同一个特征。

因此，如果喵咪特征连续一致，可以认为喵咪在运动。也就是说，如果特征<a, b>在持续帧里出现，那么它将构成特征运动。比如，特征<a, b>在第 2/3/4/7/8 帧出现，那么该特征将形成两个特征运动 2-3-4 和 7-8。

现在，给定每一帧的特征，特征的数量可能不一样。小明期望能找到最长的特征运动。

```
import java.util.*;
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){
         HashMap<String, Integer> mem = new HashMap<>();
         HashMap<String, Integer> temp_mem = new HashMap<>();
         int M = sc.nextInt();
         int max = 1;
         for(int j = 0; j < M; ++j){
            temp_mem.clear();
            int n = sc.nextInt();
            for(int k = 0; k < n; ++k){
               int x = sc.nextInt();
               int y = sc.nextInt();
               String key = String.valueOf(x) + " " + String.valueOf(y);
               temp_mem.put(key, mem.getOrDefault(key, 0) + 1);
               max = Math.max(temp_mem.get(key), max);
            }
            mem.clear();
            mem.putAll(temp_mem);
         }
         if(max <= 1){
            System.out.println(1);
         }else{
            System.out.println(max);
         }
      }

   }
}
```

小明目前在做一份毕业旅行的规划。打算从北京出发，分别去若干个城市，然后再回到北京，每个城市之间均乘坐高铁，且每个城市只去一次。由于经费有限，希望能够通过合理的路线安排尽可能的省一些路上的花销。给定一组城市和每对城市之间的火车票的价钱，找到每个城市只访问一次并返回起点的最小车费花销。

链接：

```
import java.util.*;
public class Main{
   static int get(int n,int piao[][],List<Integer> list){
      if(list.size()==1) return piao[n][list.get(0)]+piao[list.get(0)][0];
      else {
         List<Integer> templist=new ArrayList<>();
```

```
            templist.addAll(list);
            int min=Integer.MAX_VALUE,a,b=list.get(0);
            for(Integer temp:list){
                templist.remove(temp);
                a=get(temp,piao,templist)+piao[n][temp];
                templist.add(temp);
                if(a<min){
                    min=a;
                    b=temp;
                }
            }
            return min;
        }
    }
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        if(n<=20&&n>0){
            int piao[][]=new int[n][n];
            for(int i=0;i<n;i++){
                for(int j=0;j<n;j++){
                    piao[i][j]=sc.nextInt();
                }
            }
            List<Integer> list=new ArrayList<>();
            for(int i=1;i<n;i++) list.add(i);
            System.out.println(get(0,piao,list));
        }
    }
}
```

Z 国的货币系统包含面值 1 元、4 元、16 元、64 元共计 4 种硬币，以及面值 1024 元的纸币。现在小 Y 使用 1024 元的纸币购买了一件价值为[illegible]的商品，请问最少他会收到多少硬币？