题目描述：

跳房子，也叫跳飞机，是一种世界性的儿童游戏。

游戏参与者需要分多个回合按顺序跳到第 1 格直到房子的最后一格，然后获得一次选房子的机会，直到所有房子被选完，房子最多的人获胜。
跳房子的过程中，如果有踩线等违规行为会结束当前回合，甚至可能倒退几步。

假设房子的总格数是 *count*，小红每回合可能连续跳的步数都放在数组 *steps* 中，请问数组中是否有一种步数的组合，可以让小红三个回合跳到最后一格？如果有，请输出索引和最小的步数组合(数据保证索引和最小的步数组合是唯一的)。

注意：数组中的步数可以重复，但数组中的元素不能重复使用。

输入描述：

第一行输入为每回合可能连续跳的步数，它是 *int* 整数数组类型。实际字符串中整数与逗号间可能存在空格。

第二行输入为房子总格数 *count*，它是 *int* 整数类型。

输出描述：

返回索引和最小的满足要求的步数组合（顺序保持 *steps* 中原有顺序）

补充说明：

count<=10000，3<=steps.length<=10000，-100000<=steps[i]<=100000

示例 1

输入：

```
[1,4,5,2,0,2]

9
```

输出：

```
[4,5,0]
```

说明：

示例 2

输入：

```
[1,5,2,0,2,4]

9
```

输出：

```
[5,2,2]
```

说明：

示例 3

输入：

```
[-1,2,4,9]
12
```

输出：

```
[-1,4,9]
```

```
import java.util.*;

public class Main{
    public static void main(String[] args) {
        Scanner scanner    = new Scanner(System.in);
        String s = scanner.nextLine();
        s = s.substring(1, s.length() - 1);
        String [] arr = s.split(",");
        int m = 0;
        try {
            m = scanner.nextInt();
        }catch (Exception e) {
            e.printStackTrace();
        }
        int n = arr.length;
        int [] p = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = arr[i].trim();
            p[i] = Integer.parseInt(arr[i]);
```

```
        }
        Map<Integer, List<Integer>> mp = new HashMap<>();
        for (int i = 0; i < n; i++) {
            List<Integer> now = mp.getOrDefault(p[i], new ArrayList<>());
            now.add(i);
            mp.put(p[i], now);
        }
        int maxx = (int)1e9;
        int a = 0, b = 0, c = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n ; j++) {
                int need = m - p[i] - p[j];
                int flag = - 1;
                List<Integer> now = mp.getOrDefault(need, new ArrayList<>());
                for (int k = 0; k < now.size(); k++) {
                    if (now.get(k) != i && now.get(k) != j) {
                        flag = now.get(k);
                        break;
                    }
                }
                if (flag != -1 && maxx > i + j + flag) {
                    maxx = i + j + flag;
                    if (flag > j) {
                        a = i;
                        b = j;
                        c = flag;
                    } else if (flag < i) {
                        a = flag;
                        b = i;
                        c = j;
                    } else {
                        a = i;
                        b = flag;
                        c = j;
                    }
                }
            }
        }
        System.out.printf("[%d,%d,%d]", p[a],p[b],p[c]);
    }
}
```