1、多多鸡有 N 个魔术盒子（编号 1～N），其中编号为 i 的盒子里有 i 个球。
多多鸡让皮皮虾每次选择一个数字 X（1 <= X <= N），多多鸡就会把球数量大于等于 X 个的盒子里的球减少 X 个。
通过观察，皮皮虾已经掌握了其中的奥秘，并且发现只要通过一定的操作顺序，可以用最少的次数将所有盒子里的球变没。
那么请问聪明的你，是否已经知道了应该如何操作呢？
解析：要用最少的次数把所有盒子减到 0，第一次必然是减少中间盒子的球数
比如 1，2，3，4，5， 第一次减 3 得到 1，2，0，1，2 ,这时我们可以看到左右两边相等的，分治求解

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();
        for (int i = 0;i<num;i++){
            int n = sc.nextInt();
            System.out.println(cal(n));
        }
    }
    public static int cal(int n){
        if (n==1) return 1;
        if (n==2) return 2;
        else return 1+cal(n/2);
    }
}
```

2、数列 {An} 为 N 的一种排列。
例如 N=3，可能的排列共 6 种：
1，2，3
1，3，2
2，1，3
2，3，1
3，1，2
3，2，1
定义函数 F:
其中|X|表示 X 的绝对值。
现在多多鸡想知道，在所有可能的数列 {An} 中，F(N)的最小值和最大值分别是多少。
解析：意思是：在{An}的所有排列中，能让 F(N)取得的最大最小值为多少。
每四个数 例如 5，6，7，8，我们把它们两两一组 |||8-6|-7|-5|=0，最小值是 0；猜测最小值的变化也是 4 个一组，看到 min 只有 2 种取值。0，1，最大值自然就是 N-getmin(N-1)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int nums = sc.nextInt();
```

```
        for (int i = 0; i<nums; i++){
            int N = sc.nextInt();
            maxandmin(N);
        }
    }
    public static void maxandmin(int N){
        if (N==1||N==2){
            System.out.println("1 1");
            return;
        }
        //之后每 4 个一组 0011
        int min = getmin(N);
        int max = N-getmin(N-1);
        System.out.println(min + " " + max);
    }
    public static int getmin(int N){
        int temp = (N-2)%4;
        if (temp==1 || temp==2){
            return 0;
        }
        else return 1;
    }
}
```

3、多多鸡打算造一本自己的电子字典，里面的所有单词都只由 a 和 b 组成。每个单词的组成里 a 的数量不能超过 N 个且 b 的数量不能超过 M 个。多多鸡的幸运数字是 K，它打算把所有满足条件的单词里的字典序第 K 小的单词找出来，作为字典的封面。

解析：dp[n][m]表示 n 个 a，m 个 b 的单词数量

dp[n][m] = 1 + dp[n-1][m] + 1 + dp[n][m-1]

根据 K 倒推，是前半部分，还是后半部分，来确定第一个字母是 a，还是 b

注意 dp[n][m] 可能超过 long 类型的范围，所以，用 BigInteger 来存 dp

```
import java.util.*;
import java.math.*;
public class Main{
    public static void main(String[] args){
        BigInteger[][] dp = new BigInteger[50][50];
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int M = sc.nextInt();
        long K = sc.nextLong();
        for(int i=0;i<=N;i++){
            dp[i][0] = new BigInteger(Integer.toString(i));
        }
        for(int i=0;i<=M;i++){
            dp[0][i] = new BigInteger(Integer.toString(i));
```

```
        }
        for(int i=1;i<=N;i++){
            for(int j=1;j<=M;j++){
                //dp[i][j] = 1+dp[i-1][j] + 1+ dp[i][j-1];
                dp[i][j] = dp[i-1][j].add(dp[i][j-1]).add(new BigInteger("2"));
            }
        }
        StringBuilder sb = new StringBuilder();
        int n = N, m = M;
        long k = K;
        while(k>0){
            if(n>0 && m>0){
                if(dp[n-1][m].compareTo(new
BigInteger(Long.toString(k-1)))>=0){//k<=dp[n-1][m]+1
                    k--;
                    sb.append('a');
                    n--;
                }else{ //k>dp[n-1][m]+1
                    k -= dp[n-1][m].longValue()+2;
                    sb.append('b');
                    m--;
                }
            }else if(n>0 && m==0){
                k--;
                sb.append('a');
                n--;
            }else if(n==0 && m>0){
                k--;
                sb.append('b');
                m--;
            }else{
                k=0;
            }
        }
        System.out.println(sb.toString());
    }
}
```

4、在一块长为n，宽为m的场地上，有n m个1 1的单元格。每个单元格上的数字就是按照从1到n和1到m中的数的乘积。具体如下

n = 3, m = 3

1	2	3
2	4	6
3	6	9

给出一个查询的值k，求出按照这个方式列举的的数中第k大的值v。

例如上面的例子里，
从大到小为(9, 6, 6, 4, 3, 3, 2, 2, 1)
k = 1, v = 9
k = 2, v = 6
k = 3, v = 6
...
k = 8, v = 2
k = 9, v = 1
解析：import sys
def findMNK(m, n, k):
    #M*N 的矩阵，其数值范围在 1 到 M*N 之间,题目给出长为 n，宽(高)为 m,即矩阵为 m 行 n 列
    left, right = 1, m*n
    while left < right:#循环跳出条件为 left==right，区间左闭右开寻找
        mid = (left+right)//2#二分取中间值
        #求 N*M 的矩阵中有 cnt 个元素小于等于 mid
        #矩阵每一行的数据可以表示为[1*i,2*i,3*i,...n*i]
        cnt = 0
        #for i in range(1, m+1):
            #cnt += min(mid//i, n)
        #简化上续循环次数
        row = mid//n#根据矩阵的排列，可以直接求出前 row 行的数均小于 mid
        cnt += row*n
        for i in range(row+1, m+1):
            cnt += mid//i
        if cnt<k:
            left = mid+1
        else:
            # 对于 cnt==k 的情况，此时的 mid 可能不在二维表中，但是此时 mid 一定比正确答案大
            right = mid
    return left
ls = [int(i) for i in sys.stdin.readline().split()]
n, m, k = ls[0], ls[1], ls[2]
k = m*n-k+1#转化为求第 m*n-k 小的数
print(findMNK(m, n, k))