计算最接近的数

题目描述：给定一个数组X和正整数K，请找出使表达式X[i] - X[i + 1] - ... - X[i + K - 1]结果最接近于数组中位数的下标i，如果有多个i满足条件，请返回最大的i。
其中，数组中位数：长度为N的数组，按照元素的值大小升序排列后，下标为N/2元素的值

补充说明：1. 数组X的元素均为正整数；
2. X的长度n取值范围：2<= n <= 1000；
3. K大于0且小于数组的大小；
4. i的取值范围：0 <= i < 1000；
5.题目的排序数组X[N]的中位数是X[N/2]。

示例1
输入：[50,50,2,3],2
输出：1
说明：1、中位数为50：[50,50,2,3]升序排序后变成[2,3,50,50]，中位数为下标4/2=2的元素50；
2、计算结果为1：X[50,50,2,3]根据题目计算X[i] - ... - X[i + K - 1]得出三个数0（X[0]-X[1] = 50 - 50）、48（X[1]-X[2 = 50 -2）和 -1（X[2]-X[3] = 2 - 3），其中48最接近50，因此返回下标1。

```
import java.util.*;


public class Solution {
    /**
     * 语句转换
     * @param scores int 整型一维数组 分数
     * @param K int 整型
     * @return int 整型
     */
    public int findTheStartPosition (int[] scores, int K) {

        //复制数组用于排序查找中位数
        int[] sortArr = scores.clone();
        Arrays.sort(sortArr);

        int len = sortArr.length;
        //得到中位数
        int mid = sortArr[len/2];

        //定义窗口函数边界
        int l = 0;
        int r = K-1;
        //窗口函数的表达式计算值
        int sum =scores[0];
        //初始化窗口
        for (int i = 1; i <= r; i++) {
            sum -= scores[i];
        }

        //定义返回坐标
        int res =0;
```

```
        //存储遍历中找到的最小距离
        int minDiff = Math.abs(sum -mid);

        //移动窗口，算出最接近中位数的坐标
        l++;r++;
        while(r<len){
            //窗口移动时，表达式变迁公式
            sum += scores[l] * 2 -scores[l-1] - scores[r];
            //比较并更新最小距离
            int curDiff = Math.abs(sum - mid);
            if(minDiff >= curDiff){
                res =l;
                minDiff = curDiff;
            }
            l++;r++;
        }
        return res;
    }
}
```