Mad Science Quarterly
==========

The deadline for submitting papers to the Mad Science Quarterly is approaching. Professor Boolean's topic: On the Rate of Growth of Zombie Rabbits. In the lab, Boolean's minions recorded the net growth of the number of zombits for each day, which is the number of births minus the number of deaths (yes, zombits do die). He realized that if these figures were to be added up over time, it would seem like the zombits multiplied very quickly. Then everyone shall be convinced of his mad genius! He would proudly display these figures in his paper. Since some of the figures may be negative, and larger numbers are more convincing, Professor Boolean will choose which figures to show so that the sum is maximized. However, he must show a sequence of consecutive figures without omitting the unfavorable ones - Professor Boolean also needs to be scientific, you see. Unfortunately, the Mad Science Quarterly limits how much data can be shown - it is, after all, Mad. This means the mad doctor can display no more than a certain number of figures.

Write a function answer(L, k) which returns the maximum sum Professor Boolean can obtain by choosing some consecutive figures from his data. L will be a list of integers representing his data, the daily net growth of the number of zombits over a period of time. k will be the maximum number of figures he can display. Each element of L will have absolute value no greater than 100. L will contain at least 2 and no more than 7000 elements, and at least one element will be positive. k will be an integer, at least 3 and no greater than the length of L.

Languages
=========

To provide a Python solution, edit solution.py
To provide a Java solution, edit solution.java

Test cases
==========

Inputs:
    (int list) L = [-100, 95, 86, 47]
    (int) k = 3
Output:
    (int) 228

Inputs:
    (int list) L = [40, 91, -68, -36, 24, -67, -32, -23, -33, -52]
    (int) k = 7
Output:
    (int) 131

```
//Author: Neil VonHoltum
package com.google.challenges;

public class Answer {
    public static int answer(int[] L, int k) {

        int max = 0, firstklist = 0;

        for(int i = 0; i < k; i++){

            firstklist += L[i];
        }

        do{

            int currentlist = firstklist, head = 0, tail = k - 1;

            if(currentlist > max){

                max = currentlist;
            }

            while(++tail < L.length){

                if((currentlist = currentlist - L[head++] + L[tail]) > max){

                    max = currentlist;
                }
            }

            //generate next firstklist.
            firstklist = firstklist - L[--k];

        } while(k > 0);

        return max;
    }
}
```