Problem:
You are given a N x M grid. Each of the cell holds a value. You can collect the values on your path. Your aim is to go from cell (0,0) to cell (N-1,M-1). From each cell, you can only go right or down. What is the maximum summation of values you can get from your path?

What You Need To Do:
You must work independently. Complete the template submission file. This file is called MountainClimbing.java. Do **NOT** change the name of the file.
Your main function is titled mountainClimbing()**.** Do not use any arguments in this function. The auto-grader will import and then run this function for unit tests. You can create other functions as you need them, but make sure that everything is run from mountainClimbing().

This function must:

- Read in the input file. It will be titled "input" (no extension) and will contain the information listed in the **Input** section below. There is an example one in this folder. Assume it is in the same directory as your MountainClimbing.java file.

- Calculate the output for every test case, and return it as an ArrayList of Integers.

One of the tests is a timing test, which a purely recursive solution will not pass; keep this in mind when you design your solution.

Submit your file to the Gradescope assignment. You will see whether you passed each of the test cases. You can resubmit as many times as you want if you are missing any test cases. You can also look at the leaderboard to see if your code is fastest!

Input
Your input will be a text file. It will consist of:

First line: T (0 < T < 10) - the number of test cases
For every test case:
Next Line: N M - an integer value for N and an integer value for M.
Next N Lines: This will be the grid. Each line contains M integer values separated by a space.

Output
Output the maximum summation of values in your path for each case. Do this as an ArrayList<Integer>:
[output_one, output_two, output_three]

Input file (text is colored to make lines clear)	Your Output (returned by running mountainClimbing())
2 2 3 2 5 -2 1 1 1 2 3 2 5 -2 10 -3 1	[9, 10]