Description

An adorable, but lazy, quokka is descending through the branches of a tree, eating leaves on it's way home to sleep. The tree is represented by a matrix w wide and h high. Each position is an integer, representing how tasty the leaves are. These integers can be negative, because some of the leaves have bugs on.

Start point:

The quokka always starts in the top right corner of the tree (i.e. position (w, h)). This corner of the tree will *always* have a value of 0.

End point:

The quokka wants to go home to it's nest at the bottom left corner of the tree (i.e. position $(1, 1)$).

Jumping down:

The quokka can jump directly down to a new row. It can skip zero or more rows when it does this, but it will always eats the leaf it lands upon.

Hopping left:

The quokka can hop to the left, one step at a time, eating *every* leaf it finds, even if it is not tasty (it's hungry!) Because it is a *lazy* quokka and has tiny legs, it quickly gets tired – each *consecutive* step to the left incurs a score penalty:

- The first step left costs 1 point
- The second consecutive left costs 2 points
- The third consecutive left costs 3 points
- etc.

Happily, it enjoys jumping so much that whenever it moves down to a new branch it forgets how tired it's little feet were getting. Therefore the *next* time it decides to move left, the penalty cost will be back down to just 1 point again.

Other directions

The quokka *never* moves right or upwards, because it is sleepy and wants to get home quickly.

Objective

Use *dynamic programming* to calculate the maximum happiness that the quokka can achieve at the end of it's journey from the top right to the bottom left. The quokka's happiness is defined by the sum of the leaves it ate, minus the penalties accumulated from hopping left along the branches.

Example

5	-2	-1	5	3	-99	4	0
5	-2	-1	-3	3	-99	2	-3
-98	-98	-98	-98	-98	-98	-98	-98
-99	-99	-99	-99	-99	-99	-99	-99
5	0	-3	5	-1	7	-2	2
1	2	7	0	0	1	-1	-1

This example tree has dimensions $h = 6, w = 8$. The leaves eaten by the quokka on an optimal path through the tree are shaded. Note that it dropped directly from the 2 to the -2, skipping the -98 and -99, because it fell past them, but it ate the -2 leaf because it landed on it. It also ate the -1 leaf on the second row, because it cannot skip leaves when moving left. The score for this path can be calculated like this:

action	leaf	penalty	cumulative happiness
left	4	-1	3
down 1	2		5
down 3	-2		3
left	7	-1	9
left	-1	-2	6
left	5	-3	8
down 1	0		8
left	7	-1	14
left	2	-2	14
left	1	-3	12

The answer here is 12, because that is the happiness at the end of this (optimal) path.

Tasks

Task 1: [10 marks]

Define the subproblems needed by your algorithm (i.e. what does the function $OPT(...)$ represent, and what are its parameters.)

Task 2: [20 marks]

Define the recurrence(s) and base case(s) (i.e. give the mathematical functions $OPT(...) =?$)

Task 3: [30 marks]

Justify the correctness of your algorithm. In particular, take care to prove that

- the recurrence is correct
- the base case(s) are correct and sufficient

Task 4: [20 marks]

State and explain the complexity of your algorithm (in big-Oh notation). Consider both:

1. Time complexity
2. Space complexity

Make sure you justify the answers you give, don't merely state the values.

Task 5: [20 marks]

Implement the first algorithm (to calculate the value only) (on Ed), in the programming language of your choice.

Your program will read the input as a series of lines sent as standard input (i.e. as if the user had typed them at the console)

- The first line is w, the width of the tree
- The second line is h, the height of the tree
- The remaining h lines each have w integers representing the leaves (values separated by a single space each).
- The rows are input from the bottom up (i.e. the top of the tree is the *last* line of input.)

The example above would look like this:

```
8
6
1 2 7 0 0 1 -1 -1
5 0 -3 5 -1 7 -2 2
-99 -99 -99 -99 -99 -99 -99 -99
-98 -98 -98 -98 -98 -98 -98 -98
5 -2 -1 -3 3 -99 2 -3
5 -2 -1 5 3 -99 4 0
```

Your program should output a single integer, the maximum score that can be attained by the quokka.

Some skeleton code is provided in Python which reads the input for you. You can use this code, or start from scratch in this or another language if you prefer.

Please note that the test cases will not cover any weird edge cases around the input format itself (i.e. the input will always be valid, have the expected number of lines, all the numbers will be positive integers, with the specified whitespace etc.)

There will be limits on both *time* and *space* used by your algorithm. So try to keep your implementation reasonably efficient in both. If you get the "killed" message in Ed, it probably means your submission used too much memory – check with a tutor if you're in doubt of what caused this particular error message. If your submission used too much time, or gave the wrong answer, you should get a clearer feedback message from Ed.

Submission details

- Submission deadline is Friday 11th May, at 23:59pm. Late submissions without special consideration will be subject to the penalties specified in the first lecture (25% per day or part thereof.)
- Submit your answers to Tasks 1 – 4 as a single document (preferably a pdf or doc file) to Canvas. Your work must be typed text (no images of text, although you can use diagrams if you think it helps.) Please try to be reasonably concise.
- Submit your code for Task 5 to `edstem.org`
- Both your code and report will be subject to automatic and manual plagiarism detection systems. Remember, it's acceptable to discuss high level ideas with your peers, but you should not share the detail of your work, such as (but not limited to) parts of the actual algorithms, (counter)examples, proofs, writing, or code.
- COMP3927: You will probably have an additional question, we'll make that available separately early in the week. The answer to this question should be submitted separately to the rest of your report, also on canvas (to make it easier for us to coordinate the marking of that question separately.)