Tetris programming challenge

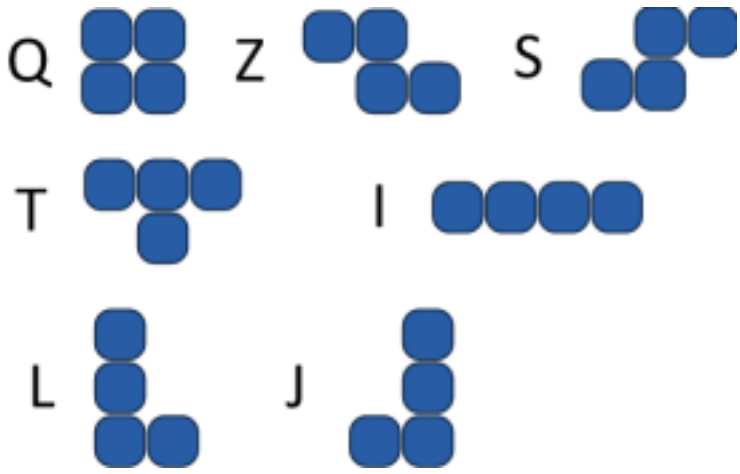
Challenge description

You are to write a simplified Tetris engine.

The engine should model a grid that pieces enter from top and come to rest at the bottom, as if pulled down by gravity. Each piece is made up of four unit squares. No two unit squares can occupy the same space in the grid at the same time. The pieces are rigid, and come to rest as soon as any part of a piece contacts the bottom of the grid or any resting block. As in Tetris, whenever an entire row of the grid is filled, it disappears, and any higher rows drop into the vacated space without any change to the internal pattern of blocks in any row.

Your program must process a text file of lines each representing a sequence of pieces entering the grid. For each line of the input file, your program should output the resulting height of the remaining blocks within the grid.

The file denotes the different possible shapes by letter. The letters used are Q, Z, S, T, I, L, and J. The shapes of the pieces they represent are shown in the diagram below.



You do not have to account for shape rotation in your model. The pieces will always have the orientations shown above.

Each line of the input file is a comma-separated list. Each entry in the list is a single letter (from the set above) and a single-digit integer. The integer represents the left-most column of the grid that the shape occupies, starting from zero. The grid of the game space is 10 units wide. For each line of the file, the grid's initial state is empty.

For example, if the input file consisted of the line "Q0" the corresponding line in the output file would be "2", since the block will drop to the bottom of the initially empty grid and has height two.

Sample input and expected interface

Your program does not need to validate its input and can assume that there will be no illegal characters

An input file called "input.txt" has been provided. Your program will be invoked from a command line, taking its input from STDIN and writing its output to STDOUT. For instance:

```
$ ./tetris < input.txt > output.txt
```

Important! - Your program will be run against an automated test-suite. It must conform to the STDIN/STDOUT interface described above

Evaluation criteria

The solution will be evaluated on the following criteria:

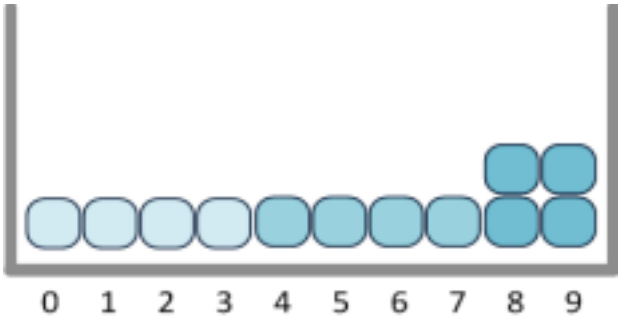
- Correctness - does the program produce the correct output
- Clarity - how easy it is for others to understand and work with your code
- Extensibility - how easy would it be to extend your program with additional functionality e.g. rotation, additional movement, new pieces, and others
- Algorithmic complexity - how does the performance of the submission scale with regards to its input
- Efficiency - how efficient is the solution. Our test suite includes test cases that might not fit entirely into memory. The solution is expected to handle multi-gigabyte inputs without running itself out of memory.

Examples

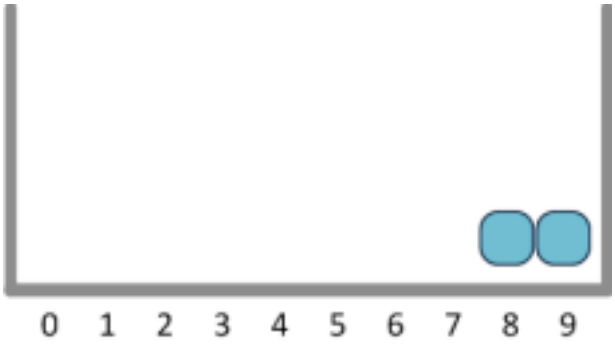
Example 1

A line in the input file contains “I0,I4,Q8” resulting in the following

configuration.



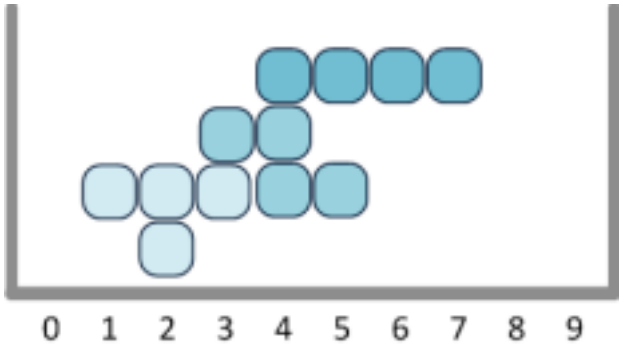
The filled bottom row then disappears.



Therefore, the output row for this sequence is “1”.

Example 2

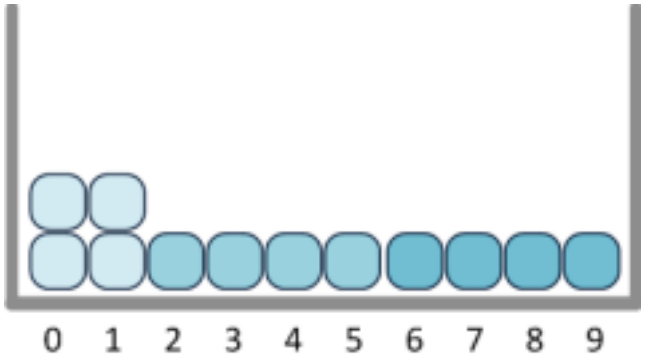
A line in the input file contains “T1,Z3,I4”.



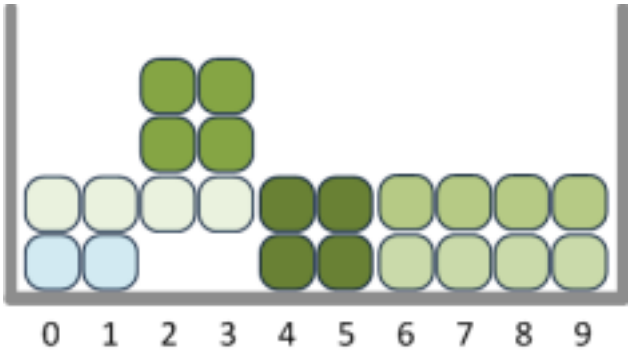
No rows are filled, so the output for this sequence is “4”.

Example 3

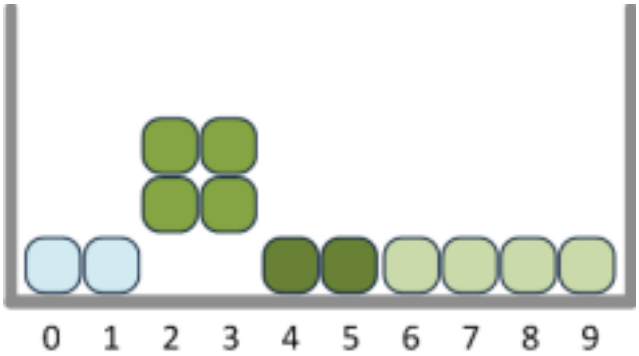
A line in the input file contains “Q0,I2,I6,I0,I6,I6,Q2,Q4”. After the first three pieces drop, the result is as follows:



The bottom line is cleared, and after the next five pieces drop, here is the result:



The second line clears, and the final result is as follows:



Note that the rows drop as rows, and do not fill gaps in the rows below. So the final output for this test case is “3”.