Scala
COMP 4210
Spring 2021

Project 1

Write a Scala program to solve the Falling Apples problem from the 2016 archives of the ACM-ICPC Mid-Central USA Regional Programmming Contest. The precise input and output format is specified, and your output must match byte-for-byte. Your program must consist of one source file called "apples.scala".

The archives linked above not only have the original problem specification, which appears on the next page, but they also have several different judges solutions, written in Python, Java, and C++. You might find these solutions helpful and insightful, but you also might find them confusing, distracting, and unhelpful; so look at them at your own discretion.

Your goals for this first project should be prioritized as follows:

1. A working solution that produced the correct output for the two sample inputs provided on the next page, and also for any other inputs that satisfy the problem specifications.

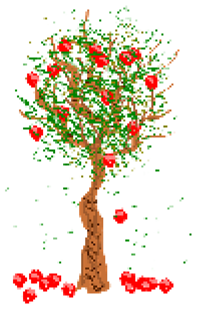
2. Your program should be written in a Scala style rather than a Java style. In other words, you should try to avoid `var`, `while`, and other imperative idioms and attempt to use functional and Scala-specific constructs and coding patterns. *However, you should not get carried away with this. Too much of anything is bad, especially when you're learning.*

3. Formatting: your code should be readable, have clarity, and be well-documented. It should also have your name at the top in comments.

You may feel free to use Scala features we haven't yet covered. However, all code you submit should be your own. Do not look at or in any other way use other students' code.

DUE: Wednesday, February 10 by 11:59 pm

Problem B
Falling Apples



You have a 2D rectangular grid. Each grid cell contains either an apple, an obstacle, or is empty. Empty cells are denoted as '`.`', apples as '`a`', and obstacles as '`#`'. You are to implement a simulation of gravity, based on the following rules:

- The obstacles do not move.
- Whenever there is an empty cell immediately below an apple, the apple moves into the empty cell.

Print out the final configuration of the board after all apples reach their final locations. Merely iterating the gravity rule, a step at a time, will likely take too long on large datasets.

Input

The input begins with a line containing integers R and C, designating the number of rows and columns of the grid, such that $1 \leq R \leq 50\,000$ and $1 \leq C \leq 10$. The first line is followed by R additional lines, each designating a row of the grid, from top to bottom. Each line has C characters, each of which is either '`.`', '`a`', or '`#`'.

Output

Output R grid lines displaying the final state.

Sample Input 1

```
3 3
aaa
#..
..#
```

Sample Output 1

```
a..
#.a
.a#
```

Sample Input 2

```
4 5
aaa.a
aa.a.
a.a..
...a.
```

Sample Output 2

```
.....
a....
aaaa.
aaaaa
```