33rd Annual Rowan University Programming Contest

hosted by the
Computer Science Department

Friday, 3 May 2019

Contest Problem

1 Introduction

Many computer games involve rectangular grids of tokens, sometimes represented as marbles, or pieces of candy, or crates, which disappear when certain conditions are met. The remaining tokens, if any, then fall down, creating a new arrangement. In some games, new tokens are added as the game progresses.

For example, in *Tetris*, the blocks disappear when a complete horizontal row is made, regardless of color. In games such as *SameGame* and *Candy Swipe*, tokens disappear when any contiguous grouping of the same category is chosen by the user. In games such as *Candy Crush* and *Move the Box*, straight-line groupings of some size disappear automatically. We will consider variations on this last idea.

A game board will be a retangular grid of tokens, represented by single-digit numbers between 1 and 9, inclusive. (A zero will represent an empty location.)

A straight-line grouping will be a vertical or horizontal set of matching tokens (*i.e.,* all the same number), of some given size. Any grouping that size or larger will automatically be removed.

A player's move is sliding a single token one square up, down, left, or right. If the token's new position already has something there, the two swap places.

Here are some sample boards of size 5x4, where the automatic deletion length is three. Different tiles are represented here by numbers, empty spaces are represented by hyphens:

```
- - - - -
- - - - -
- - 1 - -
- 1 1 - -
```

In the game shown at left, if the player moves the topmost 1 to the right, it will fall to the row below. There will then be a group of three 1 tokens, which will disappear. (This corresponds to Sample Input #1.)

Note that the player could move the topmost 1 to the left twice, and it would fall, making a group of three that would disappear. The same result, but it would take two turns.

Alternatively, the player could move the one in the center of the bottom row one space to the right, and the tile above it would fall between the other two. (The tile on top would *not* move with the one below it.)

```
- - - - -
- - - - -
- 2 1 1 -
- 1 2 2 -
```

In the game shown at left, if the player moves the leftmost 1 up, or the leftmost 2 down, then the leftmost tiles will change places. After that, there will be a group of three 1 tokens, and a group of three 2 tokens, both of which will disappear.

This position also has alternative solutions: for example, the player could move the middle 2 up, and then move the rightmost 1 down. (This corresponds to Data Set 1 for Sample Input #2.)

To indicate positions, we will number the board's rows and columns starting at zero, with (0,0) the lower-left position.

2 Problem Requirements

The challenge for this contest is to read in information about a grid in a tile-based game with automatic deletion, read in a series of moves, and print out how the board changes after each move is made.

In some cases, a move will cause tiles to fall, creating a group, causing a deletion, causing more tiles to fall. To earn full points, your program will have to follow any such sequence to its conclusion, where no more falls or deletions can occur.

3 Things To Consider

3.1 Move Specifications

```
- 2 - - - - -
- - - - - - -
- - - - - - 3
1 - - - - - -
```

All boards have position (0, 0) in the lower left. All locations are indicated by column and then row. The '1' is at (0,0). The '2' is at (1,3). The '3' is at (6,1). (If this board were the start position for a game, the '2' and '3' would both fall.)

```
- - - - - - -
- - - - - - -
- - - 1 2 - -
- 1 1 2 2 - -
```

Moves will be indicated by four numbers, indicating two positions: column, row of the tile to be moved, and then column, row of the location it is moved to.

'4 1 4 2' means to move the uppermost '2' up one. It will then fall back to its current position and nothing else will happen.

'4 1 5 1' means to move the uppermost '2' right one. It will then fall, completing a group of three '2's.

A tile can be moved one space, either vertically or horizontally. If a tile is moved to an empty space, it simply changes location. If a tile is moved to an occupied space, the two tiles swap position. For a move where (1) there is no tile to move in the first position, (2) the move is diagonal, (3) the move is more than one space, or (4) the start or end is off the board, print a message, and continue processing.

3.2 Automatic Deletion For Longer Groups

If game rules specify that the minimum group size for automatic deletion is four, any straight-line grouping of four *or more* will have all tiles removed.

From this start position:

```
- - - 2 - - -
- - - 1 - - -
- - - 2 - - -
- - - 2 - - -
1 1 1 2 1 1 1
```

The top tiles in the center are swapped:

```
- - - 1 - - -
- - - 2 - - -
- - - 2 - - -
- - - 2 - - -
1 1 1 2 1 1 1
```

The '2's vanish, the '1' falls:

```
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
1 1 1 1 1 1 1
```

There are more than four '1's; they vanish:

```
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
```

3.3 Automatic Deletion For Overlapping Groups

If a single tile appears in two separate straight-line groupings, each of which is as long as the removal threshold, both sets of tiles will be removed. (This example has a threshold of 3.)

From this start position:

```
- - - 2 - - -
- 2 2 4 4 3 4
- 3 3 2 5 6 5
- 3 3 2 3 6 5
```

The '3' and '4' in the top right are swapped:

```
- - - 2 - - -
- 2 2 4 4 4 3
- 3 3 2 5 6 5
- 3 3 2 3 6 5
```

The '4's vanish, the '2' falls:

```
- - - - - - -
- 2 2 2 - - 3
- 3 3 2 5 6 5
- 3 3 2 3 6 5
```

Both groups of three '2's will vanish:

```
- - - - - - -
- - - - - - 3
- 3 3 - 5 6 5
- 3 3 - 3 6 5
```

3.4 Cleared Board

It is possible that a board will be completely cleared, with no tiles remaining to fall, when there are still moves unprocessed. Your program should print a message and proceed to the next data set, if any, on the input. (See Section 5, 'Output', for exact details.)

4 Input

4.1 Input Specification

For text input, your program should accept input in the following format:

1. An integer, $\mathcal{D}$, where $1 \leq \mathcal{D} \leq 100$, which is the number of datasets in this file.
2. $\mathcal{D}$ data sets, each of which is in this format:
   (a) One line with two integers, $\mathcal{W}$ and $\mathcal{H}$, separated by a single space, where $1 \leq \mathcal{W}, \mathcal{H} \leq 20$, indicating the width and height of the board.
   (b) One line with one integer, $\mathcal{K}$, where $2 \leq \mathcal{K} \leq 5$, which is the minimum number of tiles that must be in a straight line for the tiles to disappear.
   (c) One line with one integer, $\mathcal{R}$, where $0 \leq \mathcal{R} \leq 50$, which is the number of rows with tiles already in them.
   (d) $\mathcal{R}$ lines, each with $\mathcal{W}$ numbers, separated by single spaces, representing the tiles already in place. If this number is greater than $\mathcal{H}$, the excess rows are tiles that will fall into the game area as the board is cleared. If this number is less than $\mathcal{H}$, the game area is not completely full. The tiles are listed from the first row with at least one tile in it, down to the bottom row.
   (e) One line with one integer, $\mathcal{M}$, where $0 \leq \mathcal{M} \leq 10$, which is the number of moves your program needs to process.
   (f) $\mathcal{M}$ lines, each with 4 numbers, separated by single spaces, representing a move to be processed.

4.2 Sample Input #1

Data in file	Item #	Meaning in plain English
`1`	`1`	*This file has 1 data set.*
`5 3`	`2.a`	*Data Set 1 has a 5x3 board.*
`3`	`2.b`	*The Data Set 1 tile elimination size is 3.*
`2`	`2.c`	*Data Set 1 has two rows with tiles filled in.*
`0 0 1 0 0` `0 1 1 0 0`	`2d`	*The position of the tiles on grid #1*
`1`	`2.e`	*Data Set 1 has one move to process.*
`2 1 3 1`	`2.f`	*The moves to process for Data Set 1.*

(This input, which corresponds to the first board position on page 2, is on the website as **sample1.txt**.)

You may choose to have your program read the input from the keyboard, or ask the user for a filename and then read the file. Users of GUI-based programming environments may prefer to use text boxes into which the values can be entered, and buttons to begin their calculation. Any reasonable variation in the spirit of the problem is acceptable.

You need not do error-checking on the input. Each line will have exactly the number of items described with no stray characters or extra spaces.

All sample and test data sets are available at <http://elvis.rowan.edu/rupc/2019>

5 Output

5.1 Output Specification

For each data set configuration, your program must generate output as follows:

1. The text '`Analyzing D data set(s)`', where D is the number of data sets in the input.
2. For each data set, print the following:
   (a) '`Data Set D`', where D is the number of the data set being reported on.
   (b) '`Board Position:`', followed by the board's initial position as read in, using single-digit integers separated by single spaces.
   (c) Groupings that begin with a line reading '`After Move` $\mathcal{N}$`:`', where $0 <= \mathcal{N} <= M$. (Move zero is for processing the initial board position, which may have tiles that disappear or fall.) Each group should contain the following:
      i. If the move meets any of the conditions in the last paragraph of Section 3.1, print `'Move is invalid.'` and go on to the next move. If the move is valid, print the new board position, and then the items below.
      ii. One or more entries of the form:
         A. Either '`No eliminations.`', if nothing is removed, or '`Board after eliminations:`', followed by the board's arrangement after any groupings have been removed.
         B. Either '`No drops.`', if nothing will drop, or '`Board after drops:`', followed by the board's arrangement after any tiles have dropped.

      It is possible that, after a tile falls, a new grouping is created that will be eliminated, and more falls will occur. Your program should process **all** falls and eliminations for a given position, and print both 'No eliminations.' and 'No drops.' before going on to the next move. One or more moves may remain unprocessed after the board is cleared.
      iii. If the board is cleared after a move, print '`The board is cleared!`' and stop looking for drops or eliminations. If there are still moves to process for that data set, print '`Skipping N move(s).`', where `N` is the number of extra moves.

5.2 Sample Output #1 (broken across two columns)

```
Analyzing 1 data set(s)
Data Set 1
 Board Position:
   - - - - -
   - - 1 - -
   - 1 1 - -

  After move 0:
   No drops.
   No eliminations.

  After move 1:
   - - - - -
   - - - 1 -
   - 1 1 - -
     Board after drops:
     - - - - -
     - - - - -
     - 1 1 1 -

     Board after eliminations:
     - - - - -
     - - - - -
     - - - - -

  The board is cleared!
```

This output corresponds to Sample Input #1, on the facing page.

6 Sample #2

6.1 Sample Input #2

```
2
5 4
3
2
0 1 2 2 0
0 2 1 1 0
1
1 1 1 0
5 3
3
3
0 0 3 0 0
0 2 2 3 0
1 1 1 2 3
1
1 1 1 0
```

6.2 Sample Output #2

```
Analyzing 2 data set(s)
Data Set 1
 Board Position:
   - - - - -
   - - - - -
   - 1 2 2 -
   - 2 1 1 -

  After move 0:
   No drops.
   No eliminations.

  After move 1:
   - - - - -
   - - - - -
   - 2 2 2 -
   - 1 1 1 -

   No drops.
   Board after eliminations:
   - - - - -
   - - - - -
   - - - - -
   - - - - -

 The board is cleared!
```

(continued next column)

```
Data Set 2
 Board Position:
   - - 3 - -
   - 2 2 3 -
   1 1 1 2 3

  After move 0:
   No drops.
   Board after eliminations:
   - - 3 - -
   - 2 2 3 -
   - - - 2 3

   Board after drops:
   - - - - -
   - - 3 3 -
   - 2 2 2 3

   Board after eliminations:
   - - - - -
   - - 3 3 -
   - - - - 3

   Board after drops:
   - - - - -
   - - - - -
   - - 3 3 3

   Board after eliminations:
   - - - - -
   - - - - -
   - - - - -

 The board is cleared!
Skipping 1 move(s).
```

Notes:

1. Data Set 1 on this input corresponds to the second example on page 2.
2. Data Set 2 on this input clears entirely from the start position, with no moves being processed at all.

   In the event that an input file has more than one data set, and one data sets has extra commands after the board is cleared, be certain you skip over the extra commands before reading input for the next data set.

Except for the board, your output **does not** have to duplicate the sample output exactly with respect to spacing or use of upper/lower case. The board should have three blanks between characters, each line indented two spaces more than the line above.

7 Sample #3

7.1 Sample Input #3

```
2
9 7
4
15
5 5 2 5 2 7 7 2 2
7 7 5 6 7 5 6 5 6
3 4 2 1 1 5 7 4 1
4 7 1 6 7 1 5 3 4
2 7 3 5 4 2 1 5 3
3 5 1 4 3 6 5 7 1
2 4 5 2 2 4 4 5 2
2 5 5 7 7 4 3 6 7
2 3 4 1 3 4 6 2 6
5 1 5 2 6 3 1 2 5
1 6 4 4 7 5 5 3 6
4 3 1 5 2 7 3 5 6
5 6 2 4 7 1 4 1 1
2 6 2 1 7 1 5 4 2
6 3 3 4 7 4 7 6 7
8
0 0 1 0
1 1 2 1
1 1 2 2
1 1 4 1
4 4 6 4
7 7 3 3
4 2 4 1
6 6 5 6
3 2
3
2
1 1 1
2 0 2
0
```

Note about Data Set 2: drops happen *before* eliminations.

The pieces settle into position before checking to see if anything can be removed.

7.2 Sample Output #3

```
Analyzing 2 data set(s)
Data Set 1
 Board Position:
   2 3 4 1 3 4 6 2 6
   5 1 5 2 6 3 1 2 5
   1 6 4 4 7 5 5 3 6
   4 3 1 5 2 7 3 5 6
   5 6 2 4 7 1 4 1 1
   2 6 2 1 7 1 5 4 2
   6 3 3 4 7 4 7 6 7

  After move 0:
    No drops.
    No eliminations.

  After move 1:
   2 3 4 1 3 4 6 2 6
   5 1 5 2 6 3 1 2 5
   1 6 4 4 7 5 5 3 6
   4 3 1 5 2 7 3 5 6
   5 6 2 4 7 1 4 1 1
   2 6 2 1 7 1 5 4 2
   3 6 3 4 7 4 7 6 7

    No drops.
    No eliminations.

  After move 2:
   2 3 4 1 3 4 6 2 6
   5 1 5 2 6 3 1 2 5
   1 6 4 4 7 5 5 3 6
   4 3 1 5 2 7 3 5 6
   5 6 2 4 7 1 4 1 1
   2 2 6 1 7 1 5 4 2
   3 6 3 4 7 4 7 6 7

    No drops.
    No eliminations.

  After move 3:
   Move is invalid.
```

(Continued in next column.)

```
  After move 4:
   Move is invalid.

  After move 5:
   Move is invalid.

  After move 6:
   Move is invalid.

  After move 7:
   2 3 4 1 3 4 6 2 6
   5 1 5 2 6 3 1 2 5
   1 6 4 4 7 5 5 3 6
   4 3 1 5 2 7 3 5 6
   5 6 2 4 7 1 4 1 1
   2 2 6 1 7 1 5 4 2
   3 6 3 4 7 4 7 6 7

    No drops.
    No eliminations.

  After move 8:
   2 3 4 1 3 6 4 2 6
   5 1 5 2 6 3 1 2 5
   1 6 4 4 7 5 5 3 6
   4 3 1 5 2 7 3 5 6
   5 6 2 4 7 1 4 1 1
   2 2 6 1 7 1 5 4 2
   3 6 3 4 7 4 7 6 7

    No drops.
    No eliminations.


Data Set 2
 Board Position:
   1 1 1
   2 - 2

  After move 0:
    Board after drops:
   1 - 1
   2 1 2

    No eliminations.
```

8 Test Data

The contest website, <http://elvis.rowan.edu/rupc/2019>, has four test data files: **test1.txt**, **test2.txt**, **test3.txt**, and **test4.txt**.

Run your program on those files and print the results. **You must submit printed output to earn full points.** Your program will also be run on data known only to the judges.

Id: rupc2019.tex,v 1.8 2019/05/02 14:31:21 kilroy Exp kilroy