CPS506 - Comparative Programming Languages - Fall 2017

Snakes and Ladders Game

Frequently Asked Questions
Course Management Form
Resources
Assignments
Tweet
Follow @RyersonCPS506

The application is a simple Snakes and Ladders game, with a few twists. The following parameters will apply:

1. The input is a series of lines, each containing one command. A command is a keyword followed by one or more parameters separated by a single space.
    1. `board 3 4` command:
    specifies the number of columns and rows for the board. The total number of cells cannot exceed 999.
    2. `players 2` command:
    specified the number of players, who are named: A, B, ... There can be up to 26 players.
    3. `dice 1 2 2 2 2` command:
    specifies the sequence of dice rolls. The sequence will repeat indefinitely - e.g. the example above would produce the sequence 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, ...
    4. `ladder 5 11` command:
    creates a ladder that starts at the first number and ends at the second number - i.e. if the player lands on the first cell they are transported to land on the second cell.
    5. `snake 8 4` command:
    creates a snake that starts at the first number and ends at the second number - i.e. if the player lands on the first cell they are transported to land on the second cell.
    6. `powerup` *`type cell list`* command
    describes a powerup that is applied to a series of cells. When a player lands on a powerup cell they acquire that powerup and retain it until they use it. Using the powerup removes it from the player. A powerup cell can be triggered any number of times by any player, but they do not accumulate - a player either has the powerup or they don't.
        1. `powerup escalator 6 9` sub-command:
        makes the next ladder cell a player steps onto twice as boosting - i.e. they move twice as far up the board. If the boost takes them past the end of the board, they get moved to the last cell, and hence win.
        2. `powerup antivenom 7` sub-command:
        make the player immune to the next snake cell they step onto - i.e. they don't slide down the snake.
        3. `powerup double 5` sub-command:
        doubles the next dice roll.

7. `turns 10` command:
   plays the specified number of turns (or until a player wins the game). A turn means each player, in order, rolls the dice, and then moves that many cells (or double if they have the powerup). If the (possibly doubled) roll would take them past the end of the board, they don't move, and play proceeds to the next player. As soon as a player wins, the game is over and the turns stop.

2. The last cell is the winning cell. If a player lands on it, by any means, they win.
3. There can be any number of snake, ladder, or powerup commands, and hence any number of the resulting game features. There can also be any number of turn commands, each of which will run in turn.
4. A given cell can only have a single special property: winning, snake start, ladder start, or powerup. Note that the *end* of a snake or ladder could have a special property.
5. A given cell can only have one player on it. When a player lands on a cell (including initial positioning), if there is already a player on the cell, that player gets bumped one cell. When a bumped player lands on a cell, they get the action associated with that cell, including winning, powerups, snakes, ladders, or bumping yet another player.
6. Assume the input is perfectly legal, with no invalid commands, extra spaces, invalid numbers, etc. Additionally assume the set-up board will not produce any bump loops during play.
7. In addition to any tests or test data we provide, you must have unit tests for the components of your program, verifying that they perform correctly. This will be worth 20% of the mark.

Sample input:

```
board 3 4
players 2
dice 1 2 2 2 2
ladder 5 11
snake 8 4
powerup escalator 6 9
powerup antivenom 7
powerup double 4
turns 10
```

Resulting output:

```
+---+---+---+
| 12| 11| 10|
|B  |   |   |
+---+---+---+
|  7|  8|  9|
| a |  S| e |
+---+---+---+
|  6|  5|  4|
| e |  L|Ad |
+---+---+---+
|  1|  2|  3|
|   |   |   |
+---+---+---+
Player B won
```

1. If no player has won, that last line would not appear at all.
2. The cell numbering starts at the bottom left, and loops back and forth.

3. Lines between rows are drawn as sequences of `+` and `-` characters, as shown.
4. Lines between columns are drawn with `|` characters, as shown.
5. Cells are printed in 2 lines. The first line is the cell number, right justified with spaces. The second line has: the player or blank, followed by the first letter of the powerup or blank, followed by the start of a snake or ladder, or blank.
6. The output must be exactly as shown, as automatic comparison will be part of marking.

Dave Mason