Assignment 3

COMP9021, Session 1, 2015

Aims: The purpose of the assignment is to:

- practice the use of stacks;
- practice performing careful computations;
- develop problem solving skills.

Submission

Your program will be stored in a file named `maze.py`. Upload your file using WebCMS. Assignments can be submitted more than once: the last version is marked. Your assignment is due by June 7, 11:59pm.

Description and expected outputs

Find out about *simple, alternating, transit mazes* at

`http://www.math.stonybrook.edu/~tony/mazes/index.html`

The purpose of the assignment is to find out whether a sequence of numbers is a *level sequence*, and if it is, output some Latex code that can be processed by pdflatex to produce a pdf file that depicts a maze determined by that level sequence. For the second task, you will probably find that the information about "the level sequence of an s.a.t. maze", accessible from the url above, is particularly useful.

Here are some possible interactions that demonstrate all possible outputs when the input is incorrect. Note that for any incorrect input, **the first of these messages that applies** should be output.

```
$ python
...
>>> from maze import *
>>> maze = Maze('0 1 2 3 4 A')
The input should consist of numbers.
>>> maze = Maze('0 1 2')
The input should consist of between 4 and 20 numbers.
>>> maze = Maze('0 1 2 3 5')
The input should consist of 0, 1, 2... N, in some order.
>>> maze = Maze('1 0 2 3 4')
The input should start with 0 and end with the largest number.
>>> maze = Maze('0 1 2 4 3 5')
The input should alternate between even and odd numbers.
>>> maze = Maze('0 3 2 5 4 1 6')
The input defines overlapping pairs.
```

Here are some possible interactions with correct inputs.

```
$ python
...
>>> from maze import *
>>> maze = Maze('0 1 2 3')
>>> maze.generate_latex_code('maze_1')
>>> maze = Maze('0 3 2 1 4')
>>> maze.generate_latex_code('maze_2')
>>> maze = Maze('0 5 4 1 2 3 6')
>>> maze.generate_latex_code('maze_3')
>>> maze = Maze('0 7 6 5 4 3 2 1 8')
>>> maze.generate_latex_code('maze_4')
>>> maze = Maze('0 9 8 5 6 7 4 1 2 3 10')
>>> maze.generate_latex_code('maze_5')
>>> maze = Maze('0 7 6 5 4 1 2 3 8 11 10 9 12 13 14')
>>> maze.generate_latex_code('maze_6')
>>> maze = Maze('0 1 2 3 4 5 6 7 8 9 14 13 12 11 10 15')
>>> maze.generate_latex_code('maze_7')
```

These statements produce files named `maze_1.tex`, `maze_2.tex`, ..., `maze_7.tex`, that compiled with pdflatex, produce the files `maze_1.pdf`, `maze_2.pdf`, ..., `maze_7.pdf`.[1]

Assessment

Incorrect input and appropriate error messages will be automatically tested for a maximum mark of 4. The Latex code produced by correct input will be automatically tested for a maximum mark of 5. Up to 1 mark will reward good readability of the source code, good comments when needed, and reasonable complexity of the underlying logic. The output has to be **exactly** as expected. Use `diff` to check whether your `.tex` files are identical to the sample `.tex` files. For instance, execute

```
$ python
...
>>> from maze import *
>>> maze = Maze('0 1 2 3')
>>> maze.generate_latex_code('my_maze_1')
>>> ^D
$ diff my_maze_1.tex maze_1.tex
```

`diff` should exit silently.

[1]For instance, the file `maze_1.pdf` is produced by executing `pdflatex maze_1.tex`