1 General presentation

You will design and implement a program that will

- analyse the various characteristics of a *maze*, represented by a particular coding of its basic constituents into numbers stored in a file whose contents is read, and
-
  - either display those characteristics
  - or output some Latex code in a file, from which a pictorial representation of the maze can be produced.

The representation of the maze is based on a coding with the four digits `0`, `1`, `2` and `3` such that

- `0` codes points that are connected to neither their right nor below neighbours
- `1` codes points that are connected to their right neighbours but not to their below ones: ─
- `2` codes points that are connected to their below neighbours but not to their right ones: │
- `3` codes points that are connected to both their right and below neighbours: ┌

A point that is connected to none of their left, right, above and below neighbours represents a *pillar*: ●

Analysing the maze will allow you to also represent:

- *cul-de-sacs*: ✕
- certain kinds of *paths*: - - -

2 Examples

2.1 First example

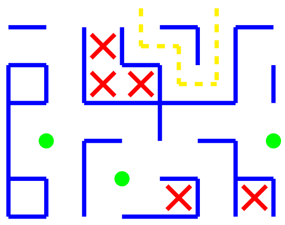
The file named `maze_1.txt` has the following contents.

```
1  0  2  2  1  2  3  0

3  2  2  1  2  0  2  2

3  0  1  1  3  1  0  0

2  0  3  0  0  1  2  0

3  2  2  0  1  2  3  2

1  0  0  1  1  0  0  0
```

Here is a possible interaction:

```
$ python3
...
>>> from maze import *
>>> maze = Maze('maze_1.txt')
>>> maze.analyse()
The maze has 12 gates.
The maze has 8 sets of walls that are all connected.
The maze has 2 inaccessible inner points.
The maze has 4 accessible areas.
The maze has 3 sets of accessible cul-de-sacs that are all connected.
The maze has a unique entry-exit path with no intersection not to cul-de-sacs.
>>> maze.display()
```

The effect of executing `maze.display()` is to produce a file named `maze_1.tex` that can be given as argument to `pdflatex` to produce a file named `maze_1.pdf` that views as follows.



2.2 Second example

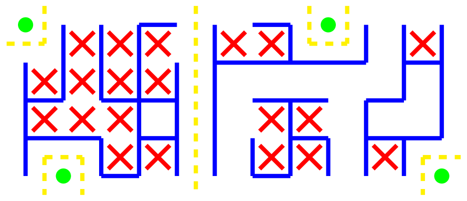
The file named `maze_2.txt` has the following contents.

```
022302120222
222223111032
301322130302
312322232330
001000100000
```

Here is a possible interaction:

```
$ python3
...
>>> from maze import *
>>> maze = Maze('maze_2.txt')
>>> maze.analyse()
The maze has 20 gates.
The maze has 4 sets of walls that are all connected.
The maze has 4 inaccessible inner points.
The maze has 13 accessible areas.
The maze has 11 sets of accessible cul-de-sacs that are all connected.
The maze has 5 entry-exit paths with no intersections not to cul-de-sacs.
>>> maze.display()
```

The effect of executing `maze.display()` is to produce a file named `maze_2.tex` that can be given as argument to `pdflatex` to produce a file named `maze_2.pdf` that views as follows.



2.3 Third example

The file named `labyrinth.txt` has the following contents.

```
31111111132
21122131202
33023022112
20310213122
31011120202
21230230112
30223031302
03122121212
22203110322
22110311002
11111101110
```

Here is a possible interaction:

```
$ python3
...
>>> from maze import *
>>> maze = Maze('labyrinth.txt')
>>> maze.analyse()
The maze has 2 gates.
The maze has 2 sets of walls that are all connected.
The maze has no inaccessible inner point.
The maze has a unique accessible area.
The maze has 8 sets of accessible cul-de-sacs that are all connected.
The maze has a unique entry-exit path with no intersection not to cul-de-sacs.
>>> maze.display()
```

The effect of executing `maze.display()` is to produce a file named `labyrinth.tex` that can be given as argument to `pdflatex` to produce a file named `labyrinth.pdf` that views as follows.