{EPITECH}
LEARN DIFFERENT*

A-MAZE-D

ELEMENTARY PROGRAMMING IN C

* apprendre autrement

A-MAZE-D

binary name: amazed
language: C
compilation: via Makefile, including re, clean and fclean rules
Authorized functions: read, write, malloc, free, getline

- ✓ The totality of your source files, except all useless files (binary, temp files, objfiles,…), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

> Your mechanical champion is ready to face off in the arena, but it needs something more: intelligence to navigate through the complex mazes of our virtual world.
>
> Welcome to the “A-maze-d” project, where you’ll need to equip your robot with the crucial ability of navigation. Picture your robots as intrepid explorers, plunged into a mysterious labyrinth, seeking the fastest path to victory.
>
> You’ll delve into the exciting world of pathfinding algorithms and graph theory. Every decision you make, every algorithm you implement, will determine the success of your robot in this intricate world.
>
> You’re embarking on an adventure that tests not only the power of your algorithms but also your creativity and strategic thinking. This is a crucial step before the grand Corewar tournament, where only the combination of raw power and strategic intelligence will lead you to victory.
>
> Are you ready to send your robot into the unknown? To equip it with the skills needed to overcome obstacles and emerge victorious? The labyrinth awaits, engineer. Leave your mark on its digital trails.
>
> **— Narrator —**

The project

Objectives

Your aim is to **move all the robots in a maze from the entrance to the exit**, while getting as many robots through as possible at the same time and in as little time as possible. To find out about the maze, your program **will receive a description of the maze on the standard input** screen.

The description will contain:
- The number of robots at the entrance to the labyrinth
- All rooms and their positions
- Tunnels linking each room

The aim of the project is first to analyze the terrain configuration to determine whether it is valid, and then to find the shortest route(s) to the exit.

Skills

In this project, you'll be working on several important skills, from data structures to algorithms:

- ✓ **Reading standard input**
- ✓ Parsing
- ✓ Graphs (adjagency)
- ✓ Matrices (adjagency)
- ✓ **Graph theory**

Delivery

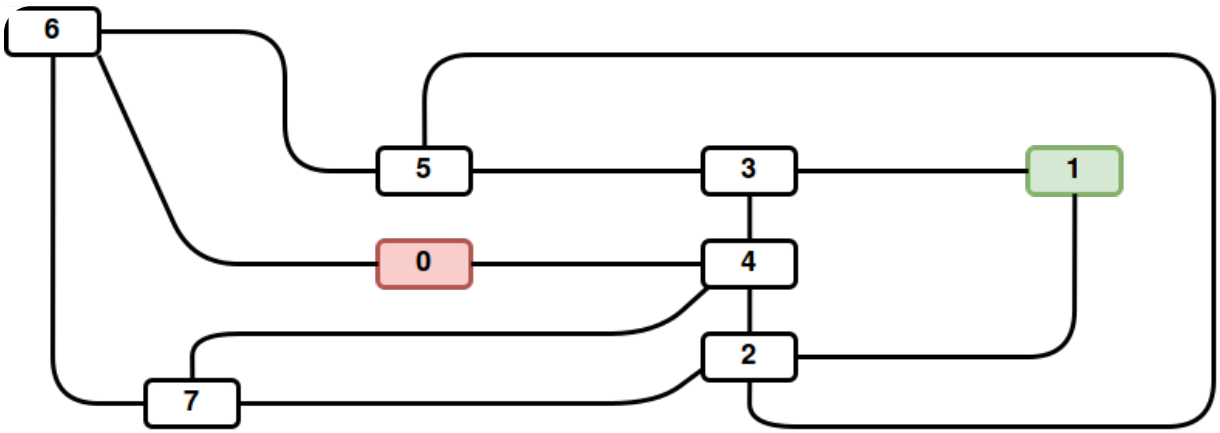
Input

Your program must be able to read the description of your labyrinth from its standard input, which can be generated using a **Perl script provided in the project attachments**. The parameters of the script are the number of rooms, the density of connections and the number of robots.

Terminal

```
B-CPE-200> ./laby_gen.pl 8 15 34 > input && cat input
34
##start
1 23 3
2 16 7
#comment
3 16 3
4 16 5
5 9 3
6 1 0
7 4 8
##end
0 9 5
0-4
0-6
1-3
4-3
5-2
3-5
#another comment
4-2
2-1
7-6
7-2
7-4
6-5
```

The first non-conforming or empty line will terminate the acquisition of the labyrinth, as well as its normal processing with the data already acquired.



The names of the rooms won't necessarily be numbers in order, and are less likely to be uninterrupted (you could even have rooms named: zdfg, 256, qwerty, …)

The rooms coordinates will always be whole numbers. Please note that it is possible to insert comments by using "#" and commands by using "##".
##start indicates the next room is the labyrinth entrance, and ##end indicates the next room is the labyrinth exit.
Any unknown commands will be ignored.

Output

The objective of your program is to find the quickest way to make the robots cross over the labyrinth. To do so, each single robot need to take the shortest route (and not necessarily the easiest), whithout walking on its peers, and avoiding traffic jams.

At the beginning of the game, all of the robots are in the labyrinth entrance.
The goal is to lead them to the exit room, **in a minimum amount of laps**.
Each room could contain a single robot at a time (except ##start and ##end, which can contain as many as needed).
With each lap, you can move each robot only once by following a tunnel (if the receiving room is clear).

You must show the result on the standard output, in this order: `number_of_robots`, `rooms`, `tunnels` and then for each lap, a series of `Pn-r` where n is the number of the robot, and r the name of the room it gets into. In the output, you must display a comment indicating the part that will follow. These comment must be exactly like in the examples.

It's not as easy as snapping your fingers. Concerning what type of operation students of magic can perform with such a computer, all we know, today, is that electricity is more dependable.

Examples

Concerning the following trivial anthill:
[0] - - - [2] - - - [3] - - - [1]

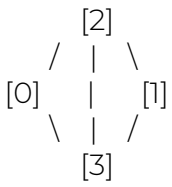
Terminal

```
B-CPE-200> cat labyrinth
3
##start
0 1 0
##end
1 13 0 #bedroom
2 5 0
# The next room is the kitchen
3 9 0
0-2
2-3
3-1
```

Terminal

```
B-CPE-200> ./amazed < labyrinth
#number_of_robots
3
#rooms
##start
0 1 0
##end
1 13 0
2 5 0
3 9 0
#tunnels
0-2
2-3
3-1
#moves
P1-2
P1-3 P2-2
P1-1 P2-3 P3-2
P2-1 P3-3
P3-1
```

There you have it, it's over after 5 laps.



Terminal

```
B-CPE-200> cat labyrinth
3
2 5 0
##start
0 1 2
##end
1 9 2
3 5 4
0-2
0-3
2-1
3-1
2-3
```

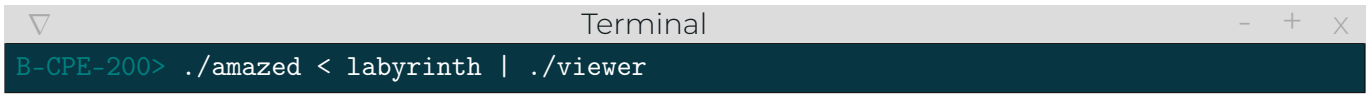
Terminal

```
B-CPE-200> ./amazed < labyrinth
#number_of_robots
3
#rooms
2 5 0
##start
0 1 2
##end
1 9 2
3 5 4
#tunnels
0-2
0-3
2-1
3-1
2-3
#moves
P1-2 P2-3
P1-1 P2-1 P3-2
P3-1
```

It's over after 3 laps.

Bonus

For a bonus, why not code an labyrinth viewer?
Whether in two dimensions (an overhead view) or from an robot's point of view in a 3D corridor environment (using the Oculus Rift?).
Please note that because the commands and comments are repeated on the output of the program, it is possible to pass specific commands to the viewer (why not colors, levels, …?).
An example of usage may be:



v1

* apprendre autrement