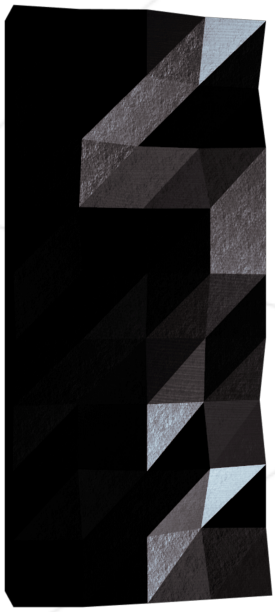


# Starfleet Academy

## Grimly

gaetan gaetan@42.fr
femi femi@42.us.org

*Summary: Will you find the exit?*

# Contents

# Chapter I

# Foreword

Here's what the Guide has to say about traveling in space:

« The Infinite Improbability Drive is a new method of crossing vast interstellar distances without all that tedious mucking about in hyperspace. As soon as the drive reaches Infinite Improbability, it passes through every conceivable point in every conceivable Universe almost simultaneously, then selects the appropriate re-entry point. The principle of generating small amounts of finite improbability by hooking the logic circuits of a Bambleweeny 57 Sub-Meson Brain to an atomic vector plotter suspended in a strong Brownian Motion producer, say a nice hot cup of tea, were of course well understood.

Such generators were often used to break the ice at parties by making all the molecules in the hostess's undergarments leap simultaneously one foot to the left, in accordance with the Theory of Indeterminacy.

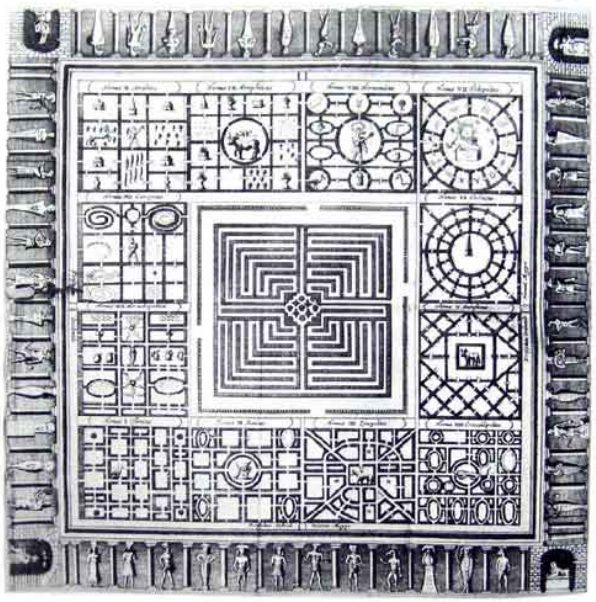
Many respectable physicists said they weren't going to stand for that sort of thing partly because it was a debasement of science, but mostly because they didn't get invited to those sort of parties. »

# Chapter II

# Introduction

The Egyptian Labyrinth

The first recorded maze in history was the Egyptian Labyrinth. Herodotus, a Greek traveler and writer, visited the Egyptian Labyrinth in the 5th century, BC. The building was located just above Lake Moeris and opposite the city of the crocodiles (Crocodilopolis). Herodotus was very impressed by it, stating, "I found it greater than words could tell, for although the temple at Ephesus and that at Samos are celebrated works, yet all the works and buildings of the Greeks put together would certainly be inferior to this labyrinth..." Herodotus added that even the pyramids were surpassed by the Egyptian Labyrinth. [1]



reconstruction of the Egyptian Labyrinth by Athanasius Kircher. Copperplate engraving (50X 41 cm)
"Turris Babel Sive Archontologia", Amsterdam 1679

Image Source: https://egyptexperience.wordpress.com/2013/06/07/the-egyptian-labyrinth-at-hawara/

[1] Krystek, L., Amazing Mazes

# Chapter III

# Subject

- Walk in a labyrinth :
    - This project is about finding the shortest path between entering and leaving a labyrinth while avoiding obstacles.
    - A maze is given to you in a file to be passed as an argument to the program (Maze generator in Appendix).
    - The first line of the labyrinth contains the information to read the map:
        * The number of labyrinth lines then the number of columns (LINExCOL);
        * The "full" character
        * The "empty" character
        * The character "path"
        * The character "entered labyrinth"
        * The character "exit labyrinth".
    - The maze is composed of "empty" characters, "full" characters, characters "entering the labyrinth" and characters "exiting the labyrinth".
    - The purpose of the program is to replace "empty" characters with "path" characters to represent the shortest way to cross the labyrinth.
    - Movements can only be horizontally or vertically, not diagonally.
    - In the case where several solutions exist, one will choose to represent the shortest one. In case of equality, it will be the one whose exit where the solution is the most up then the leftmost.
      If there are 2 solutions for the same output, when crossing from the start we will choose the solutions in this order:
      up> left> right> down
      So if you have a choice between going up or right, you have to take the solution that goes up.

- Definition of a valid card :
  - All lines must respect the sizes given in the first line (LINExCOL).
  - There can only be one entrance.
  - There must be at least one exit.
  - There must be a solution to the labyrinth.
  - The labyrinth will not be more than a billion square.
  - At the end of each line, there is a new line.
  - The characters present in the card must be only those shown on the first line.
  - If there is an invalid card, you will see the error output: `MAP ERROR` followed by a new line. The program will then proceed to the next labyrinth treatment.

- Examples :

```
$>cat -e 01.map
10x10* o12$
***1******$
*       **$
* *  **  *$
*        *$
** ** ** *$
*       **$
*      ***$
**       *$
*        *$
***2******$
$>./grimly 01.map
10x10* o12
***1******
*  o    **
* *o **  *
* oo     *
**o** ** *
* oo    **
*  o   ***
** o     *
*  o     *
***2******
RESULT IN 10 STEPS!
$>
$>
$>cat -e 02.map
10x10* o12$
**1*******$
*      * *$
*  *   ***$
*    *   *$
*   *    *$
*        2$
* *      *$
*     *  *$
***    ***$
****2*****$
$>
$>
$>./grimly 02.map
10x10* o12
**1*******
* o    * *
* o*   ***
* oo *   *
*  o*    *
*  oo    2
* * o    *
*   o *  *
*** o  ***
****2*****
RESULT IN 10 STEPS!
$>
```

```
$>./grimly bug.map
MAP ERROR
$>
```

# Chapter IV

# Instructions

- The executable must be named `grimly` and be in the root directory.
- The root directory must be named `grimly`.
- Your project must follow the Norme.
- The directory must have an author file in which you must put your login:

```
$>cat author
login
$>
```

- The program should take several files in parameter.
- If no argument is passed, the program will read a maze from the standard input.
- Your Makefile will compile the project.
- You can only use the `open`, `close`, `write`, `read`, `malloc` and `free` functions.
- You can ask your questions on the forum.

# Chapter V

# Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.

`This project is only graded by Moulinette, hence make sure your output is identical with the examples and you print errors to stderr`

# Chapter VI

# Appendix

- Generator example.

```ruby
#!/usr/bin/env ruby

### Made By Sgregory :-)

if ARGV.count < 3 || ARGV[2].length < 5
  puts "./usage height width characters"
else
  height, width, chars, gates = ARGV[0].to_i, ARGV[1].to_i, ARGV[2], ARGV[3].to_i
  entry = rand(width - 4) + 2
  entry2 = rand(width - 4) + 2
  puts("#{height}x#{width}#{ARGV[2]}")
  height.times do |y|
    width.times do |x|
      if y == 0 && x == entry
        print chars[3].chr
      elsif y == height - 1 && x == entry2
        print chars[4].chr
      elsif y.between?(1, height - 2) && x.between?(1, width - 2) && rand(100) > 20
        print chars[1].chr
      else
        print chars[0].chr
      end
    end
    print "\n"
  end
end
```