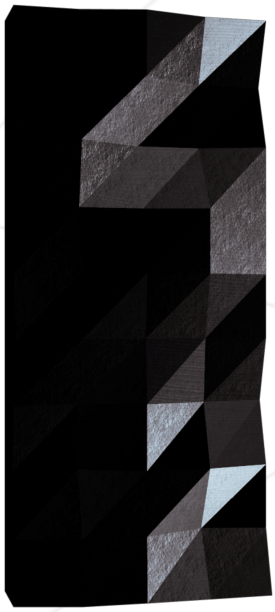

C-Piscine's final project

BSQ

Summary: Will you find the biggest square?

Version: 8.3

Contents

Chapter I

Foreword

Quotes from “Life, The Universe, and Everything”:

« Important facts from Galactic history, number one:
(Reproduced from the Siderial Daily Mentioner’s Book of popular Galactic History.)
The night sky over the planet Krikkit is the least interesting sight in the entire Universe. »

« The Krikkit Wars belonged to the ancient past of the Galaxy, and Zaphod had spent most of his early history lessons plotting how he was going to have sex with the girl in the cybercubicle next to him, and since his teaching computer had been an integral part of this plot it had eventually had all its history circuits wiped and replaced with an entirely different set of ideas which had then resulted in it being scrapped and sent to a home for Degenerate Cybermats, whither it was followed by the girl who had inadvertently fallen deeply in love with the unfortunate machine, with the result (a) that Zaphod never got near her and (b) that he missed out on a period of ancient history that would have been of inestimable value to him at this moment. »

« The game you know as cricket, [Slartibartfast] said, and his voice still seemed to be wandering lost in subterranean passages, is just one of those curious freaks of racial memory which can keep images alive in the mind aeons after their true significance has been lost in the mists of time. Of all the races on the Galaxy, only the English could possibly revive the memory of the most horrific wars ever to sunder the Universe and transform it into what I’m afraid is generally regarded as an incomprehensibly dull and pointless game. »

« Although it has been said that on Earth alone in our Galaxy is Krikkit (or cricket) treated as a fit subject for a game, and that for this reason the Earth has been shunned, this does only apply to our Galaxy, and more specifically to our dimension. In some of the higher dimensions, they feel they can more or less please themselves, and have been playing a peculiar game called Brockian Ultra-Cricket for whatever their transdimensional equivalent of billions of years is. »

Chapter II

Instructions

- Your work must comply with the Norm. If you have bonus files/functions, they will be included in the Norm check, and you will receive a score of 0 if there is a Norm error.
- You must handle errors coherently, in accordance with the instructions of this subject.
- Your project must be completed and pushed to the Git repository by the deadline displayed on the project's main page on the intranet.
- You must follow the submission procedure described at the end of this document, if provided.
- Your program must compile using `cc` with the following flags: `-Wall -Wextra -Werror`. If there is a Moulinette, it will use the same compiler and flags.
- If your program does not compile, you will receive a score of 0.
- Each member of the group must be fully aware of all the details of the project. If you choose to split the workload, ensure you understand every part completed by other team members. This understanding may be verified during the evaluation.

```
This project will be checked and graded by a program called
Moulinette.  Moulinette is very meticulous and strict in its
evaluation of your work.  It is entirely automated and there is
no way to negotiate with it.  So, to avoid any bad surprises, be as
thorough as possible.
```

Chapter III

Subject

Program Name	`bsq`
Files to Submit	`Makefile and all the necessary files`
Makefile	`Yes`
Arguments	`File(s) containing the map data`
External Function	`open, close, read, write, malloc, free, exit`
Libft authorized	`Not applicable`
Description	`Write a program that computes and prints the biggest possible square in the given area.`

- The biggest square:
  - The aim of this project is to find the biggest square on a map while avoiding obstacles.
  - A file containing the map will be provided. It will have to be passed as an argument to your program.
  - The first line of the file contains information on how to read the map:
    - The number of lines on the map; it can be any valid `int`.
    - The "empty" character.
    - The "obstacle" character.
    - The "full" character.
  - The “empty”, “obstacle” and “full” characters are the last 3 characters of the line, all other characters in front of them represent the number of lines.
  - The map is made up of lines containing "empty" characters and "obstacle" characters.
  - The task of the program is to replace some "empty" characters with "full" characters in order to represent the biggest square possible.
  - In the case that more than one solution exists, we will choose to represent the square that is closest to the top of the map, then the one that is most to the left.

  - Your program must handle 1 to n files as parameters.
  - When your program receives more than one file as an argument, there will be an empty line between two different solutions or `map error` messages.
  - If no arguments are passed, your program must be able to read a single file from the standard input.
  - You must have a valid Makefile that compiles your project. Your Makefile must not relink.

- Definition of a valid file:
  - The first line must start with a valid positive number, followed by three different printable characters and the newline character.
  - Note that the space character, or digits, can be part of the last 3 characters of the line.
  - All lines of the map must have the same length.
  - There is at least one line of at least one cell on the map.
  - Lines are separated by the usual newline character.
  - The characters on the map can only be those introduced in the first line.
- In case of an invalid file or map, your program should display `map error` on the standard output followed by a newline. Your program will then move on to the next available file, if any.
- Here's an example of how it should work:

```
%>cat example_file
9.ox
.........................
....o....................
............o............
.........................
....o....................
..............o..........
.........................
......o..............o.....
..o.......o..............
%>./bsq example_file
.....xxxxxxx.............
....oxxxxxxx.............
.....xxxxxxxo............
.....xxxxxxx.............
....oxxxxxxx.............
.....xxxxxxx...o.........
.....xxxxxxx.............
```

```
......o..............o.....
..o.......o.................
%>
```

It is a square indeed. Even though it might not look like that visually.

Chapter IV

Appendix

- Perl map generator

```
#!/usr/bin/perl

use warnings;
use strict;

die "Usage: program x y density" unless (scalar(@ARGV) == 3);

my ($x, $y, $density) = @ARGV;

print "$y.ox\n";
for (my $i = 0; $i < $y; $i++) {
    for (my $j = 0; $j < $x; $j++) {
        if (int(rand($y) * 2) < $density) {
            print "o";
        }
        else {
            print ".";
        }
    }
    print "\n";
}
```

- Test your program with many maps of different sizes. Keep various maps available for the evaluation.

Chapter V

Submission and peer-evaluation

Submit your assignment to your `Git` repository as usual. Only the work inside your repository will be evaluated during the defence. Do not hesitate to double-check the names of your files to ensure they are correct.

You need to submit only the files requested by the subject of this project.

This project will be checked and graded by a program called Moulinette. Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So, to avoid any bad surprises, be as thorough as possible.