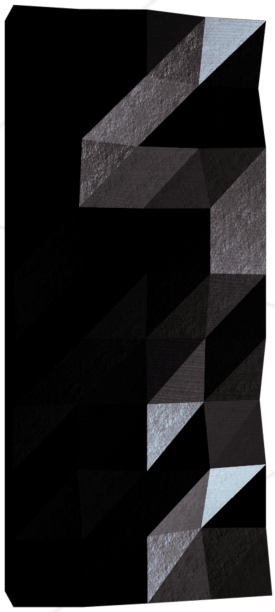

AlCu

A rush on algorithms

Summary: This rush will make you work on efficient progamming methods and algorithms.

Version: 1

Contents

Chapter I

Preamble

A marble is a small spherical object often made from glass, clay, steel, plastic, or agate. Marbles can be used for a variety of games called marbles.
The games has been played in many countries, but the rules are made up by the players, and there are many variations. One standard idea is to have a target marble. Players flick their marbles with their thumbnail, and try to hit the target.
Another version is where players try to hit each other's marbles out of a target zone.

Chapter II

Introduction

This project invites you to create a turn based strategy game.
The board consist of N heaps of any number of items.
Each player alternate taking between 1 to 3 object from the last heaps. The goal is to be the last to take an object.
Chance does not intervene. Be a `winner`.

Chapter III

General rules

- Your assignment must be written in `C`.
- No norm.
- `CC` is used as compiler.
- You have to compile your program with the following flags: `-Wall -Wextra -Werror`
- You have to turn in a `Makefile` which will compile your source files. It must not relink.
- Your `Makefile` must at least contain the rules: `NAME`, `all`, `clean`, `fclean` and `re`.
- Your program should not quit unexpectedly (segmentation fault, bus error, double free, and so forth) except for undefined behaviors. If it happens, your project will be considered non-functional and your grade will be `0`.
- Within the mandatory part, you're allowed to use the following functions:
  - `read`
  - `write`
  - `open`
  - `close`
  - `malloc`
  - `free`

Chapter IV

Project instructions

IV.1 Game instructions

- The game will be played on a board, for example, this is a board:

```
   |||||||
    |||||
     |||
      |
```

- There is a fixed number of heaps.
- There is 2 players, one of them is an AI.
- Players take turns, AI start first.
- On each player turn they can take, on a single heap, between 1 and 3 items.
- The player who takes the last item, on the last available heap, loses.

IV.2 Program instructions

- The board will be read from a file or standard input with an empty line signaling the end of input if no arguments are given.
- The board will be formatted according to the following rules:
  - Each line indicate the number of items present on that line, followed by a newline.
  - This number must be included between 1 and 10000.
  - If the board is incorrect you'll write `ERROR` on standard error, followed by a newline.
- The board must be displayed between each turn.

- On player turn, you will ask him the number of items they want to remove.
  - Items are removed from the last heap.
  - This must be a valid number.
  - In case of error, you'll ask again.
- Your AI must try to win.
- At the end of the game, the winner is announced.

Chapter V

Exemple

```
$> cat -e alcu.map
8$
5$
3$
2$
1$
$
$> ./alum1 < alcu.map
 | | | | | | | |
   | | | | |
     | | |
      | |
       |
AI took 1
 | | | | | | | |
   | | | | |
     | | |
      | |
Please choose between 1 and 3 items
3
3 - Invalid choice
Please choose between 1 and 3 items
2
 | | | | | | | |
   | | | | |
     | | |
...
You are the winner! Congratulations!
```

Chapter VI

Bonus part

The only bonus there is to provide some sort of vizualizer. You're free to use any library you want, here are some:

- Termcap
- ncurses
- MiniLibX
- SDL
- Metal
- OpenGL
- ...

The bonus part will only be accessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will be evaluated at all.

Chapter VII

Turn-in and peer-evualation

As usual, turn in your work on your repo GiT. Only the work included on your repo will be reviewed during the evaluation.

Good luck.