Introduction to Programming a.y. 2024/2025
Project

October 10, 2024

The document contains the guidelines for the project implementation.

1 Project Description

Students are required to develop a C program that implements the logic of the following card game:

1.1 Components

- A deck of 40 cards divided into 4 suits
  - 10 cards - Clubs (from 1 to 7, with J,Q K)
  - 10 cards - Spades (from 1 to 7, with J,Q K)
  - 10 cards - Diamonds (from 1 to 7, with J,Q K)
  - 10 cards - Hearts (from 1 to 7, with J,Q K)

1.2 Players

- From 2 to 20 players

1.3 Goal of the game

- Be the last player in the game

1.4 Game Play

At the beginning of the game each player has **2** LIFE POINTS.
The game is played in phases. At each stage:

1. Shuffle the deck.
2. Deal each player 2 cards: one face down and one face up.
3. Randomly choose the first player of the phase.

1.4.1 Progress of the Phase

The phase takes place by giving each player a turn. Starting from the first player, in ascending ring order, the player in the turn:

1. must resolve the effect of the face-up card (see card effect).
2. can view the face-down card and optionally reveal/flip it (from face-down to face-up) to apply its effect. If it has already been discovered due to the effect of other cards, it will not be possible to apply its effect again.

3. ends the turn and the play moves on to the next player. At the end of all players' turns, the phase ends and whoever has no more lives is eliminated from the game.

What does ascending ring order mean?

- Suppose to have 3 player in game (P1, P2, P3)
  - If the first player is P1, the phase will follow the turns P1 → P2 → P3
  - If the first player is P2, the phase will follow the turns P2 → P3 → P1
  - If the first player is P3, the phase will follow the turns P3 → P1 → P2

1.5 Card Effect

- **Cards from 2 to 6**: no effects.
- **Card 7**: the player who holds this card (face-up) forces the next player to reveal the face-down card and apply its effect.
- **Card J**: the player who holds this card (face-up) must give 1 LIFE POINT to the previous player.
- **Card Q**: the player who holds this card (face-up) must give 1 LIFE POINT to the second following player. (i.e., skipping a player).
- **Card 1**: the player who holds this card (face-up) drops 1 LIFE POINT to the playing field (the life points on the playing field are not reset at each phase, so they remain even for the following ones if they have not been claimed).
- **Card K**: the player who holds this card (face-up) must claim all the LIFE POINTS dropped until that time on the playing field.

1.6 End of the Game

The game ends when only one player remains at the end of a phase and the others have all been eliminated from the game.

1.7 FAQ - Rules of the Game

- **In two players, how to does the effect of Card Q work?** The player who holds that card will lose 1 LIFE POINT and will subsequently receive his own LIFE POINT as there is a ring order of the players.
- **At each phase, 2 cards are distributed to each player, but can these be accumulated in subsequent phases?** No. All cards at the end of each phase are collected and shuffled into the deck, then redistributed to each player in the next phase. Players will only ever have 2 cards.
- **If the current player did not reveal the card, would it have to be discarded?** No, it remains in play face down because it could be activated by other players using special effects (e.g. the last player of the turn could use the effect of Card 7, affecting the first player of that turn, potentially leading to the chain activation of other face-down card effects).

2 Specifications for the Implementation

The program will ask players which move to play, and will display the updated playing field (using the `printf` function or equivalent). It must verify that the chosen move is legal (does not conflict with the rules of the game), and must update the components in the various phases of the game. These steps will be repeated until the end of the game, which will be notified on the terminal.

In the event that the current player optionally requests to reveal/flip a card that would lead to the loss of the current player, a warning message is requested to be displayed and confirmed (es: "`The move causes you to lose all your life points. Are you sure you want to proceed? [Yes/No]`").

Furthermore, it is possible to present improvements to the project, which can be positively evaluated. Possible improvements could be: better graphics, more usable user interaction menu, etc. There are no limits to the additions you want to make! Don't set limits!

2.1 FAQ - Implementation

- **Should a specific data structure be used to implement the game?** No, it is left to the students to decide how and which data structures to use.

3 Delivery

When? The project must be submitted 4 days before the date of each exam deadline. It is always mandatory to register for "Exercises" session of prof. Olivieri. Then, the project can be discussed and evaluated.
Where? The teacher will enable delivery via Moodle according to the scheduled deadlines.
What? You will deliver a single zip file containing:

1. A written report of a maximum of 3 pages describing the structure of your project, the organization of work among the group members, the main difficulties encountered. Longer report (more than 3 pages) will be penalized.

2. The source code of the project in C99 language; any parts of code written in other dialects must be isolated in separate sources and the project must compile appropriately.

3. Documentation of functions, types and files generated with Doxygen External libraries.

4. You can implement the entire game simply using the standard C library, using functions like `scanf`, `getc` o `getchar` to read values from the users and the function `printf` to print messages on the terminal. Those who want to use external libraries to give a more captivating and fluid graphic design and interaction with the player can do so, however it will be their responsibility to take care of the portability and compatibility aspects. The program should compile and work correctly on 3 platforms (Windows, Linux, Mac); otherwise, it is necessary to appropriately document and justify the choice in the report that will accompany the project and during the examination.

4 Evaluation

The project is to be carried out in groups of **max 4 students**. The composition of the groups is free. In exceptional cases (e.g. working students, ...), the project can also be developed individually if agreed with the prof. Olivieri.

The project will be evaluated with a grade from 0 to 30, and will account for 30% of the final grade. The project **must be sufficient** (a grade greater than or equal to 18) to be able to verbalize the final grade.

The project will be evaluated indicatively with the following criteria::

Group Project

- A project is **sufficient**, if it allows at least 2 users to play by entering their moves via keyboard input (interactive mode) and displaying the playing field and their cards in the current turn on the terminal; the implementation of the rules must be complete and correct.

- A project is **good**, if it implements the previous point and implements an iterative logic that allows you to parametrically add the number of users (no 20 hard-coded users!) and the default number of players' life points (e.g. greater than 2); Special cases such as the insertion of invalid moves or values must also be handled;

- A project is **excellent**, if it correctly implements the previous points and exhaustive Doxygen documentation of the project is provided;

Everyone in the group needs to know every line of the code!
Important: The project is group, but the evaluation is individual. This means that members of a group may receive a different grade.

Individual Project (e.g., for working student)

- A project is **sufficient**, if it allows at least 2 users to play by entering their moves via keyboard input (interactive mode) and displaying the playing field and their cards in the current turn on the terminal; the implementation of the game rules can be reasonably partial.
- A project is **good**, if it implements the previous point and implements an iterative logic that allows you to parametrically add the default number of players' life points (e.g. greater than 2); Special cases such as the insertion of invalid moves or values must also be handled;
- A project is **excellent**, if it correctly implements the previous points and exhaustive Doxygen documentation of the project is provided.