ECE 175: Computer Programming for Engineering Applications

Final Project: *A Game of Trains*

Due Date: Tuesday, April 30th, 11:59 PM

1 Administrative Details

1.1 Groups

You are allowed to form groups of two or work individually. Groups of three or more are not allowed. If you decide to work as a pair, use collaborative tools to facilitate the project development, such as Zoom, GitHub, Google drive, etc. Make sure nobody else can access your code (if you are using Github, or replit).

1.2 Submission

The project is due on Tuesday, April 30, 2024 at 11:59 pm. If you work as a group, **both team members must submit the code** in the designated zybooks dropbox. Write your teammate's name on your submission. You may also submit any design documents that will help you explain your code on D2L. Late submission policy 10% deduction per day. Please turn in:

- Your .c and .h files (in case you make libraries).
- Any input files you used for running test cases.
- Any design documents that explain the structure of your code.

1.3 Academic Integrity Policy

Each team is expected to submit its own code. You may ask other for advice, and in general discuss the project, but you should WRITE YOUR OWN CODE. Violations of academic integrity will result in a 0 for the final project and a failing grade for the class. This policy will be very aggressively enforced. ALL submitted code will be checked with a plagiarism detection tool. If you use code from the notes and Zybooks, cite the source by commenting next to the copied code.

1.4 Suggestions

- Read the project carefully to understand all aspects. Ask questions if you do not understand something.
- Spend time designing your code and use modular programming. Create all the function prototypes and describe what they do before developing your code. Create pseudocode of your program flow. This allows you to divide up the code development easily with your partner.
- Determine test cases for your functions and your overall code to ensure proper functionality.
- Try to reuse functions from the lectures. Be sure to document any code used from class.
- Write well-documented code.
- Check in code frequently. Not only does this demonstrate your progress, it also is useful if you need to go back to a prior version after frequent changes. This is a standard industry practice. Since you will be working in pairs, both members should check in the code to zybooks regularly.

2 A Game of Trains

You are to develop an interactive version of Game of Trains between two players. The rules are described at https://trehgrannik.com/upload/Game_of_Trains_rules_eng.pdf . You can watch a video of the gameplay at https://www.youtube.com/watch?v=2c9DuQpgrMI . Your program should operate as follows.

2.1 Specifications

1. The cards are represented as variables of the following type:

```
typedef struct {
    int value;
    char action[15];
} card;
```

You are allowed to add attributes to this definition, but not to remove any. The action field is used to denote the function of action cards.

2. The draw pile is implemented by an array of 84 cards.
3. Each player's hand is implemented by an array of 7 cards.
4. The center face-up cards row is implemented by an array of 8 cards.
5. A sample input file is provided for loading the card deck. This will be used in testing, so ensure you can load cards from this file.

2.2 Setup

1. Prompt the user to either load in a file to preset the deck of cards, or build the deck and shuffle them for a regular game.
2. Deal seven (7) cards to each player. Sort each player's hand in descending order (highest card first).
3. Player 1 draws one card and replaces one card in their train with it. The replaced card goes into the face up card row.
4. Player 2 draws two cards and discards one to the discard pile (face down). Player 2 then plays the other card, replacing a card in their train with it. The replaced card goes into the face up card row.
5. If playing, the third and fourth players start with 3 cards and discard two of them, and plays the last one, following the same process as above.

2.3 GamePlay & Rules simplifications

The gameplay process is described at https://trehgrannik.com/upload/Game_of_Trains_rules_eng.pdf Some game simplifications are as follows.

- At the beginning of the game, the user can choose to shuffle the deck or load a predefined sequence of cards from a file (for testing). A sample file will be provided.
- There is no need to keep track of the discard pile. If the game has not ended by the time the draw pile is exhausted, the game terminates, and the winner is the one with the longest

order of train numbers **starting with the first train car**. (So, if the train cards are 20, 4, 5, 6, 12, 28, 53, 80, that player has 1 car in order.)

- Limit the game to two players.

2.4 Rules clarifications

- If you draw a card from the face-down draw pile, it must be used as the number. You must replace one the cars from your row with the new card (you cannot discard the card just drawn.)
- If you draw from the face-up center row, it must be used as the action. You cannot place the card in your train.
- If there are ever two cards in the face up row that have the same action, both cards are discarded to the discard pile. If there are more than 2 cards in the center row with the same action (possible if everyone discards cards with the same action), only discard in pairs. So, three cards would discard 2 and keep 1 in the center row. It doesn't matter which one, since only the action is important for the face up row, not the number on the card.
- As soon as someone's train is in the correct order, the game is over and they win. Even if this is on a different player's turn.

2.5 Grading

- Grading will be based upon the following criteria:
    - Following specifications
    - Functionality of the program
    - Intuitive user interface – easy to play
    - Good coding practices
        - Frequent check in of code to zybooks to demonstrate incremental development
        - Good documentation to demonstrate design

2.6 Optional Features for Extra Credit

Please note in the header of your main.c file, which of the following you implemented, so that we can check it accordingly.

1. **Personalized.** Prompt the user for the player's name. Use this name for any prompts to the player to enter in information. (2 pts).
2. **Play a full game.** This means when the draw pile is empty, take the discard pile (the face down cards), shuffle them and make a new draw pile. Play until someone wins. (3 pts)
3. **Additional players.** Allow the game to be played by any number of players from 2–4. Prompt the user for the number of players. (5 pts)
4. **Automated players.** Modify your code to allow a player to be played by the computer. You should prompt the user for which player(s) are human or computer. (5 pts)
5. **Graphics.** Add graphics to your game. You can print cards using ascii art. (5 pts)