CSC10012 PROJECT: POKER

Note: Unlike the examinations, where function and data structures are specified, in projects, they are suggestions. Students may either follow the instructions or implement them differently to create a runnable sourcecode. Requirements for projects include:

- Proposing an effective method to solve the problem,
- (Recommendation) Decomposing large problem into smaller subtasks,
- Coding consistently using a standard style guide,
- Providing a well-organized and commented source code that is easy to understand and follow.

I What is Poker?

Poker is a family of comparing card games in which players wager over which hand is best according to that specific game's rules. Today, Poker is usually played with a standard deck with 52 cards [1]. A standard 52-card deck comprises 13 ranks, from highest to lowest: `A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2`, in each of the four equal suits: `clubs (♣), diamonds (♦), hearts (♠) and spades (♥)`. In this project, we don't construct a real complete Poker game, but only use its rules to make a simple competitive version, including PvE and PvP types.

Gameplay: Basically, the dealer shuffles the cards and deals a appropriate number of cards to each player, depending on the poker variant, one at a time. Cards can be dealt face-up (community cards) or face-down (hole cards) depending on the game. After the initial deal, the first betting round begins, followed by additional rounds where players' hands may develop. This development often involves dealing additional cards (community cards) or replacing some cards in their hand (discarding and drawing). The player with the best five-card hand according to the specific poker variant wins the pot. There are 10 different kinds of poker hands, descending in order of strength: [2]:

- **Straight Flush:** A straight flush is a hand that contains five cards of sequential rank, all of the same suit, such as Q♥ J♥ 10♥ 9♥ 8♥ (a "queen-high straight flush"),
- **Four of a kind**, also known as quads or "four cards", is a hand that contains four cards of one rank and one card of another rank, such as 9♥ 9♦ 9♣ 9♠ J♥ ("four of a kind, nines"),
- **Full house** is a hand that contains three cards of one rank and two cards of another rank, such as 3♣ 3♠ 3♦ 6♣ 6♥ (a "full house, threes over sixes"),
- **Flush** is a hand that contains five cards all of the same suit, not all of sequential rank, such as K♣ 10♣ 7♣ 6♣ 4♣ (a "king-high flush"),
- **Straight** is a hand that contains five cards of sequential rank, not all of the same suit, such as 7♣ 6♥ 5♠ 4♥ 3♠ (a "seven-high straight"),
- **Three of a kind** is a hand that contains three cards of one rank and two cards of two other ranks, such as 2♦ 2♠ 2♣ K♥ 6♥ ("three of a kind, twos"),
- **Two pair** is a hand that contains two cards of one rank, two cards of another rank and one card of a third rank (the kicker), such as J♥ J♣ 4♣ 4♠ 9♥ ("two pair, jacks and fours"),
- **One pair** is a hand that contains two cards of one rank and three cards of three other ranks (the kickers), such as 4♥ 4♠ K♠ 10♦ 5♠ ("one pair, fours"),
- **High card**, also known as no pair or simply nothing, is a hand that does not fall into any other category, such as K♥ J♥ 8♣ 7♦ 4♠ ("high card, king").

Please visit Wikipedia for more information.

[1] https://en.wikipedia.org/wiki/Poker

[2] https://en.wikipedia.org/wiki/List_of_poker_hands

II Project requirements

II.1 Standard features

This mode contains the essential steps to make the game playable, including:

- initializing a simple Poker game for N players (where data structure should be designed first),
- dealing 5 cards to each player,
- displaying the game board and determining the winner,
- updating players' information, including statistical numbers such as winrate, rank, the favorite strategy (most frequently played winning hand).

Technical requirements:

- There must be a structural design for the board and player account,
- Game content must be included both dynamic and fixed array,
- Any additional information that is not necessary for the game but will be used for statistical analysis must be stored in a file (any kind).

II.2 Advanced features

Players will experience the game better if you add extra features, including:

- Variant Poker Games: Such as Draw Poker, Five-Card Stud, or others [3],
  - Draw poker: Games in which players are dealt a complete hand, hidden, and then improve it by replacing cards,
  - Five-Card Stud: Games where players receive a combination of face-up cards (typically two, unique) and face-down cards (typically three, shared amongst all players).
- PvE Version: This mode has a computer-controlled dealer that plays as an opponent. For a realistic experience, the dealer's win rate could be set slightly higher than other players,
- Color and Sound Effects: Enhance the gameplay experience,
- Leaderboard: Allows players to track their performance and compare it to others.

III Submission and Grading

III.1 Submission

Your submission **ID1_ID2_ID3.zip**, uploaded to Moodle, must contain the following files and folders:

- `Sourcecode` folder consists of all **.cpp* and other related files:
  - executable and generated using the `g++` compiler. If a different compiler is used, it must be clearly documented in the report (using a different compiler is not recommended),
  - well-structured and commented where necessary.

[3] https://en.m.wikipedia.org/wiki/List_of_poker_variants

- `report.pdf` should follow a specific structure:
  - a cover page with a table of contents,
  - **Introduction**: your name, ID and class; along with a table of your contribution, for example:

	rubric	ID1	ID2	ID3
Standard features	–			
initialization	20	10	10	
dealing	20	20		
gameplay	20	4	6	10
players' information	20			20
Advance features	–			
draw pocker	30	15	15	
5-card stud	30			30
color/sound effects	30	5	5	5
leader board	30			20
your creations	30/each			
Report	–			
	100	35	25	40
	Total	**60**	**75**	**25**

  - **Programming**:
    - a tutorial on how the game works, including visual aids like diagrams or flowcharts, as well as instructions for running the program using files and terminal commands,
    - verbal descriptions/pseudo-code for functions in your project, explaining your design and implementation,
    - your creations/extra features should be described in detail.
  - **Refenrence**: weblinks, books, or any materials used must be appropriatedly cited. If you discuss with your classmates or any high-level mentor(s), their name must be listed too.

Note: **DO NOT** include your sourcecode in the report.

III.2 Grading

You will be evaluated by your implementation, the report, and what is provided in the contribution table (scores in the rubric column are fair and equitable for every student group). So how to get zero score for this project?

- not submitting source code or a report, or
- violating regulations, or
- having an unrunnable program, or
- cheating (copy/refer without any references).

IV Any hint?

Designing data structures to save information of 52-card desk, hands, players, ...

```
struct Card {
    enum suit;
    enum rank;
};
struct Hand {
    Card cards[5];
};
struct Player {
```

```
    int id;
    string username;
    float winrate;
    int rank;
    string favorite_strategy;
};
```

Shuffling cards using random generation.

- `void shuffling_cards(Card desk [52], int shuffled_desk[52]);`
- Initialize a 52-element empty array and place neither randomly position (in shuffled desk) for each card, or vice versa, each card sequentially into a random cell

A♣	2♣	3♣	4♣	5♣	6♣
12	15	2	42	30	51

	Ace	2	3	4
♥	15	18	2	6
♦	9	3	4	7
♣	20	29	51	0
♠	50	49	33	8

Dealing cards To deal a number of cards sequentially or round-robin to every player, and then print hand(s).

- `Hand* dealing_cards(Card desk [52], int shuffled_desk[52]);`

Checking the strength of hand/Comparing hands to detect the winner

- `int check_strength_of_hand(Hand);`
- `int get_winner(Hand*);`