CSC10002 - PROGRAMMING TECHNIQUES

MIDTERM PROJECT – POINTERS

April 4, 2021

I Project Content

I.1 Card shuffling & Dealing

Consider a standard deck of 52 cards, each of which is characterized by

- Suits: {"Hearts", "Diamonds", "Clubs", "Spades"}

- Ranks: {"Two", "Three", "Four", "Five",
  "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen" "King", "Ace"}

1. Write a function that shuffles cards.
   `void shuffleCards(int deck[][])`

   - Initialize a 2-D array (matrix): `int deck[SUITS][FACES] = {0};`
     where rows are for suits and columns are for faces (see Figure 1).

   - Each element of the matrix represents the order (from 1 to 52) of a
     card in the deck. *(Source: [Deitel and Deitel, 2015])*

   - Randomly put each number in the range $[1, 52]$, to every element of
     the matrix

2. Write a function that prints out the resulting shuffling.
   `void printCardsShuffling(int deck[][], char* suits[], char* faces[])`

   - Initialize an 1-D array of four elements, *suits*:
     `char* suits[SUITS] = {"Hearts", "Diamonds", "Clubs", "Spades"};`

   - Initialize an 1-D array of thirteen elements, *faces*
     `char* faces[FACES] = {"Ace", "Two", "Three", "Four", "Five",`
     `"Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen" "King"}`
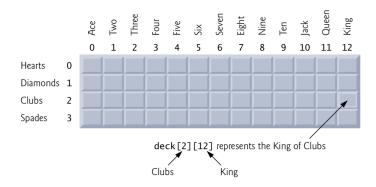
Figure 1: The matrix *deck* stores the shuffled cards.

- Prints out the card following their orders (from 1 to 52), each of which is characterized by its $(suit, face)$. For example,
$(Hearts, Four)$
$(Clubs, Eight)$
$(Clubs, Four)$

I.2 Card game: Five-card Poker

There are multiple players in a poker game. Cards are distributed circularly such that each player receives five cards.

For example, for four players,

- The first player receives the $[1, 5, 9, 13, 17]$ cards,

- The second player receives the $[2, 6, 10, 14, 18]$ cards,

- The third player receives the $[3, 7, 11, 15, 19]$ cards, and

- The fourth player receives the $[4, 8, 12, 16, 20]$ cards.

Hand-ranking categories (from best to worst) are defined as follows:

 (i) *Straight flush*: a hand that contains five cards of sequential rank, all of the same suits

 (ii) *Four of a kind or quads*: a hand that contains four cards of one rank and one card of another rank

 (iii) *Full house*: a hand that contains three cards of one rank and two cards of another rank

(iv) *Flush*: a hand that contains five cards all of the same suit, not all of sequential rank

(v) *Straight*: a hand that contains five cards of sequential rank, not all of the same suit

(vi) *Three of a kind*: a hand that contains three cards of one rank and two cards of two other ranks

(vii) *Two pairs*: a hand that contains two cards of one rank, two cards of another rank and one card of a third rank

(viii) *Pair*: a hand that contains two cards of one rank and three cards of three other ranks

(ix) None of the above categories: the highest card is taken as a representative.

The player whose hand contains five higher cards wins. If two players fall into the same category (e.g., two pairs), it is a tie.

1. Write a poker game for one player only. You may need to implement the following functions:

   a) A function that distributes cards to a player. The resulting array stores five cards assigned to the player. Each card is represented by a 2-D array containing the row and column indices of the matrix `deck`
   `int** dealingForHand(int deck[SUITS][FACES])`
   E.g., `result[i][0] = row`, `result[i][1] = column`

   b) A function that prints out five cards assigned to a player. The 2-D array `hands` stores the five cards of the player
   `void printHand(int** hand, char* suits[], char* faces[])`
   E.g., $(Clubs, Five)(Clubs, Nine)(Diamonds, Five)$
   $(Diamonds, Ten)(Hearts, Eight)$

   c) [*Optional*] A function that generates a test case (i.e., five cards for a player) for subsequent functions
   `int** createHandTest(int deck[SUITS][FACES]), int a[])`

   d) A function that checks whether a hand contains *Four of a kind*
   `int isFourOfAKind(int** hand)`

   e) A function that checks whether a hand contains *Full house*
   `int isFullHouse(int** hand)`

   f) A function that checks whether a hand contains *Flush*
   `int isFlush(int** hand)`

   g) A function that checks whether a hand contains *Straight*
   `int isStraight(int** hand)`

h) A function that checks whether a hand contains *Straight flush*
   `int isStraightFlush(int** hand)`

i) A function that checks whether a hand contains *Three of a kind*
   `int isThreeOfAKind(int** hand)`

j) A function that checks whether a hand contains *Two pairs*
   `int isTwoPairs(int** hand)`

k) A function that checks whether a hand contains *Pair*
   `int isPair(int** hand)`

l) A function that returns the value of the highest card
   `int getHighestCard(int** hand)`

2. Write a poker game for 2 players (you may want to extend for n players, $n > 2$). You may need to implement the following functions.

   a) A function that distributes cards to n players.
      `int*** dealingForHands(int deck[SUITS][FACES], int n)`

   b) A function that returns the hand-ranking of five cards (8 if there is $straight flush$, 0 if they do not fall into any hand-ranking category)
      `int getStatusOfHand(int** hand)`

   c) A function that ranks n players in one turn and returns an array of n elements such that the player i^th is in the rank $a[i]$
      `int* rankingHands(int*** hands, int n)`

   d) For s times of dealing cards, write a function that calculate the sum of scores of n players and congratulate the winner.
      `int* evaluateHands(...)`

3.* Write a poker game for *dealer* side. The dealer also receives five cards, yet he may additionally draw one, two or three cards to replace some old cards (new cards are continuously drawn from the current deck).
   The replacement of one, two, or three cards from the set of five cards can be decided following (1) random replacement or (2) replace to get better situation.

4.*** Write a program that lets a player and the dealer compete with each other. The player may decide whether to additionally draw one, two or three cards or not.

5.**** Replace the decision making algorithm of the dealer to have different game levels (easy, medium and hard)

II   Regulations

- This is a 2-person group project.

- Deadline: 20h00 - 17/4/2020.

- Your file submission must be named **Student1's ID _ Student2's ID.rar(.zip)** which is compressed from the **Student1's ID _ Student2's ID** folder. This folder includes:

    - The **Report.pdf** file. You can see the requirement contents of this file in the section below.

    - The finished **Checklist.xlsx** file.

    - **Source** sub-folder that contains your source code (*.cpp, *.h). Any other extensions submission must be declared and explained in your report.

- Plagiarism and Cheating will result in an "0" (zero) for the entire course and will be subject to appropriate referral to the Management Board of High-Quality Program for further action.

III   Evaluation

The project will be graded on your source code and report.

III.1   **Source code** *(Total: 120 pts)*

- Complete the given functions, make sure that the function prototype is correct. *(110 pts)*

- Build a user menu and displays for the features from **II.1** and **II.2** *(10 pts)*

III.2   **Report and Checklist** *(Total: 80 pts)*

- You should fulfill the checklist come within the document. (10pts)

- The report should be in English. *(10 pts)*

- The report must be presented clearly and logically. The length of your report must not exceed 10 A4 pages.

- Compulsory contents:

    - Full name and Students's ID of group's members.

- Explain briefly the idea of the given functions.

- Flowchart of your program.

- References.

References

P.J. Deitel and H.M. Deitel. *C how to Program: With an Introduction to C++*. How to program series. Pearson, 2015. ISBN 9781292111087.