Introduction to Programming, PIC10A
E. Ryu
Spring 2017

UCLA

Homework 8
Due 5pm, Wednesday, May 31, 2017

Download the starter code `main.cpp`, `card.h`, `deck.h`, `hand.h`, `shuffle.h`, and `deck.cpp`. Do not modify the header files. We have provided `main.cpp` to give you an idea of how we intend to use the functions. Put the implementations into the files `card.cpp`, `deck.cpp`, and `hand.cpp`. Other cpp files must not contain a `main` function. You may not use any libraries aside from `iostream`, `string`, and `cassert`. You may not use global variables.

We may take off up to 20% of the total marks for poor style; make sure to name your variables reasonably, indent properly, and comment sufficiently. Submit `card.cpp`, `deck.cpp`, and `hand.cpp`.

Problem 1: (Card)

Write the member function implementations for the class `card`, which simulates the standard playing card, into the file `card.cpp`.

The private data member

```
int num;
```

represents the number of the card. `1` corresponds to an Ace, `11` to a Jack, `12` to a Queen, and `13` to a King.

The private data member

```
char suit;
```

represents the suit of the card. The 4 possible suits are `'C'`, `'D'`, `'H'`, and `'S'`.

The public method

```
void setNum(int n);
```

sets the number of the card. `assert` that the input is between `1` and `13`.

The public method

```
void setSuit(char s);
```

sets the suit of the card. `assert` that the input is one of `'C'`, `'D'`, `'H'`, or `'S'`.

The public method

```
int getNum();
```

returns the number of the card. `assert` that the number is between `1` and `13`.

The public method

```
char getSuit();
```

returns the suit of the card. `assert` that the suit is one of `'C'`, `'D'`, `'H'`, or `'S'`.

The public method

```
string read();
```

simulates reading the card. The return value should look like

```
Ace of Clubs
8 of Diamonds
10 of Hearts
Queen of Spades
```

The general rule should be clear from these 4 examples.

The default constructor

```
card();
```

initializes `num` to be `0` and `suit` to be `'X'`. A default-initialized `card` represents an invalid `card` waiting to be initialized with `setNum` and `setSuit`. (Calling `getNum` immediately after default-initialization will cause `assert` to fail.)

The constructor

```
card(int n, char s);
```

initializes `num` and `suit` with the inputs. `assert` that the inputs are proper.

Remark. By making `num` and `suit` private members that must be accessed through member functions, we can do error checking. You will see why this is useful when you start using `card` in the implementations of `deck` and `hand`.

Hint. `string(1, c)` converts a `char` of name `c` into a `string`.

Hint. Starting from C++11, the `string` library provides `string to_string(int val);`, which you may find useful.

Problem 2: (Deck)

Write the member function implementations for the class `deck`, which simulates the standard deck of 52 playing cards, into the file `deck.cpp`.

The private field

```
card stack[52];
```

represents the 52 cards of a deck.

The private field

```
int deal_count;
```

represents how many cards have been dealt from the deck.

The public member function

```
void print_deck();
```

prints out the deck in order. The implementation is provided.

The default constructor

```
deck();
```

simulates preparing a brand new deck of cards. So `deal_count` is initialized to `0`, and the stack of cards are initialized so that calling `print_deck();` immediately after initialization outputs

```
Ace of Clubs
2 of Clubs
...
King of Clubs
Ace of Diamonds
...
King of Diamonds
Ace of Hearts
...
King of Hearts
Ace of Spades
...
King of Spades
```

The public member function

```
void shuffle();
```

shuffles the deck in a uniformly random order. We've provided the implementation, since it requires a random numbers, a topic we won't cover.

The public member function

```
card deal();
```

simulates dealing a card. So `deal()` sould return the `card` "at the top of the deck", and the value of `deal_count` should be incremented by `1`. When a card is dealt without shuffling, the first card should be `Ace of Clubs` and not `King of Spades`. `assert` that `deal_count` is less than `52`.

The public member function

```
int stack_size();
```

returns the current size of the deck. If no cards have been dealt, the size is `52`. If all cards have been dealt, the size if `0`.

The public member function

```
void gather_and_shuffle();
```

simulates gathering all the cards that have been dealt and then shuffling. So after `gather_and_shuffle()` is called, `deal_count` is set to `0` and the deck is shuffled.

Hint. When defining `gather_and_shuffle()`, call `shuffle()`.

Remark. Because `stack` is defined as an array, `card`'s default constructor is called. This is why `card` must have a default constructor. If you don't understand why, try removing the `card`'s default constructor.

Problem 3: (Hand)

Write the member function implementations for the class `hand`, which simulates a hand of 2 cards, into the file `hand.cpp`. The relative strength of 2 hands are determined by the following rules:

- A pair (two cards of the same number) is the strongest hand.
- Two cards of the same suit is the next strongest hand.
- Two cards of different numbers and suits is the weakest hand.
- Within the same kind of hands, the stronger hand is determqined by the larger number.
- If two hands are of the same kind and largest numbers are the same, the stronger hand is determined by the smaller number.
- If all above fails, the two hands are of equal strength. I.e., all suits are of equal strength.
- 2 is the weakest number. An Ace is stronger than a King.

You are using a single deck of cards. So a hand of A♥A♥ is impossible. Here are some examples:

- 3♣3♦ is stronger than 9♠2♠.
- 9♠2♠ is stronger than 8♦7♦.
- 8♦7♦ is stronger than 8♥6♥ .
- 8♥6♥ is stronger than A♥J♦.
- 3♣3♠ has the same strength as 3♦3♥.

The public data members

```
card c1, c2;
```

represent the 2 cards of a hand.

The public member function

```
bool operator<(hand rhs);
```

returns `true` if `rhs` is the stronger hand and returns `false` otherwise.

The public member function

```
bool operator==(hand rhs);
```

returns `true` if the two hands are of equal strength and returns `false` otherwise.

The public member functions

```
bool operator>(hand rhs);
bool operator<=(hand rhs);
bool operator>=(hand rhs);
```

are defined similarly.

Hint. Using `operator<` all other operators can be defined in one line. In doing so, you may find the `this` pointer useful.