University of Wollongong
School of Computer Science and Software Engineering

CSCI124/MCS9124 Applied Programming Autumn 2015

Lab 9 (1 mark)
Due at 11:59pm on Thurs, May 14.

Background:
In this lab you will gain more experience in designing classes.

Task One:
We first have to model a standard playing card and a deck of cards using classes.
A single card will be represented using the following class (`card.h`):

```
class card
{
     public:
          card();
          card(int);
          void set(int);
          void print(ostream&) const;
     private:
          int value;     // 0-51, -1 means uninitialised
};
```

The cards of a standard deck are stored as an integer from 0 to 51, representing the 52 cards. Numbers 0 to 12 are spades, 13 to 25 are clubs, 26 to 38 are diamonds and 39 to 51 are hearts. In each of these 13 values, the ranks of the cards are `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `X` (making 10 printable as a single character), `J`, `Q`, `K`, and `A`. The default constructor should set the value to be -1 indicating an uninitialised card. The initialising constructor should set the value to the provided value, checking for the legality of the number, generating an error and terminating the program if illegal.

The function `set()` allows for a card to be set to a specific value with appropriate checks including that the card is not being changed from an already valid value.

The `print()` function provides an output of the card as a pair of characters such as `5H` for the five of hearts, `XD` for the ten of diamonds, and `AS` for the ace of spades. There should be no other output but the two characters (except for the word `uninitialised` for an uninitialised card). Given this definition, implement `card.cpp` which represents a single card from a deck.

Now we have a class to represent a single card and perform actions on it. A single card doesn't really have that much value. Cards are grouped together to form a deck of 52. Once we have a deck, we can perform actions on it: shuffling and dealing (the top card off the deck).

In order to model a deck of cards on the computer with the above actions, we will need additional methods and information.

So we define a class called `deck` which contains 52 cards (as defined by the `Card` class above).

```
class deck
{
     public:
          . . .
     private:
          int size;
          card set[52];
};
```

We have to implement the functionality of a deck as described above, that is with methods to shuffle and deal. Shuffling appeared in an earlier lab exercise. We'll assume that the top of the deck is the last card in the set, while size is how many cards are in the deck. Initially, of course, size will be 52, and will decrement each time a card is dealt. Consider: Do we need an initialising constructor? A destructor? Place the class implementation in `deck.cpp` with the header going in `deck.h`.

While constructing both classes, use a driver program `main.cpp`. In the end, this driver should demonstrate the functionality of both `deck` and `card` by doing the following:

1. create a deck of cards.
2. shuffle the deck of cards
3. deal a poker hand of five cards from the deck of cards, printing the dealt cards using the syntax defined above.

You may want to repeat the process of dealing cards till there are no cards left in the deck. This may influence some of the functionality in the `deck` class.

The names of all files, classes and methods can be changed if you desire.

Submission:

You are to submit the five files (driver, card and deck implementation files and interface files) using the submit program:

```
$ submit –c csci124 –a lab9 –u USERNAME files
```