University of Wollongong

School of Computer Science and Software Engineering

CSCI124 **Applied Programming** **Spring 2012**

Assignment 3 **(Due: Week 8, Wednesday 12 September)** **6 marks**

Aim:

This assignment is to familiarise you with the use of classes in your programs.

On completion you should know how to:

- Write programs with class objects.
- Implement programs incrementally to minimise debugging.
- Design class objects for software applications.

Requirements:

This assignment involves the implementation of a simple class to hold sets of playing cards. This structure can be used in any card games requiring decks of cards or hands of cards. A standard deck of cards is represented by the numbers 0 to 51. Thus the contents of the class will be a (dynamic) array of integers with values in that range. These integers will represent the usual 52 cards, starting from the 2 of spades (denoted 2S) followed by 3,...,10 (denoted as XS), followed by J(ack), Q(ueen), K(ing) and A(ce). Then will follow the clubs, diamonds and hearts. To facilitate the output of a card, the file **CardSet.cpp** contains a function PrintCard(int) which will print one card. This file will also hold the implementation of the class CardSet whose interface file is provided as **CardSet.h**. The contents of this class:

```
class CardSet
{
	public:
		CardSet();
		CardSet(int);
		~CardSet();
		int Size() const;
		bool IsEmpty() const;
		void Shuffle();
		int Deal();
		void Deal(int,CardSet&,CardSet&);
		void Deal(int,CardSet&,CardSet&,CardSet&,CardSet&);
		void AddCard(int);
		void MergeShuffle(CardSet&);
		void Print() const;
	private:
		int* Card;
		int nCards;
};
```

A driver program in **main.cpp** has also been provided for testing the class Cardset. Before commencing any work familiarise yourself with these files and the task performed by each class function described below. All work is to be done in CardSet.cpp. Do not alter and submit **CardSet.h** and **main.cpp**.

Implementation:

The task for this assignment is to add the implementations of the member functions to the file **CardSet.cpp**. Here is a description of the required member functions:

(i) CardSet();

This is the (default) constructor. It should set up a set of 0 cards.

(ii) CardSet(int);

This is the initialising constructor. It should set up a set of cards, where the number of cards is the argument passed. The array Card should be filled with cards starting from 0, up to the number of cards needed, but

never exceeding the number 51. For example if a set of 104 cards is requested, the array would contain 0 to 51 twice.

(iii) ~CardSet();
The destructor should clean up any dynamic memory used.

(iv) int Size() const;
This accessor function should return the value of nCards.

(v) bool IsEmpty() const;
This function should return true if there are no cards in this set.

(vi) void Shuffle();
This function rearranges the cards in the set in a random manner. There are many ways of doing this. The simplest method follows this algorithm:

```
FOR each card i in the set
    SET j to be a random number on range 0 to number of cards -1
        IF i is not equal to j THEN
            exchange card i and card j
```

You should use the cstdlib library function rand() to generate the random integers needed.

(vii) int Deal();
This function should return the value of the first card in the set, located in the 0th element of the array. The function should then create fresh memory, transfer the rest of the set to this new memory, and then delete the old memory. That is, this class never has any vacant space in it. The array is always the same size as the number of actual cards it holds. If the set is empty, this function should print an error message and terminate the program.

(viii) void Deal(int,CardSet&,CardSet&);
This function deals two hands into the two CardSet arguments passed. The number of cards to be placed into each hand is the first argument. The cards should be removed from the current set one at a time, placing the cards into alternate hands. For example. if the current set held 2S, 3S, 4S, 5S, 6S, 7S (the integers 0 to 5) and we had to deal 3 cards, then the two hands would get 2S, 4S, 6S (integers 0, 2, 4) and 3S, 5S, 7S (1, 3, 5) respectively. The two hands may already have cards in them, and the additional cards will require new memory. Do not create new memory more often than is required. Remember that the current set has to be reduced in size as well. If there aren't enough cards in the current set to perform the deal, print an error message and terminate.

(ix) void Deal(int,CardSet&,CardSet&,CardSet&,CardSet&);
Same as above but for four hands.

(x) void AddCard(int);
Function to add a card to the current hand.

(xi) void MergeShuffle(CardSet&);
This function takes the current set and the set provided as an argument and makes the current set contain all the cards from the two sets, with cards alternating from each set as far as possible. After this function the argument set will be empty.

(xii) void Print() const;
This accessor function should use PrintCard to print out the contents of the set five cards to a line.

Commence this assignment by providing stubs for all the above functions so that the program will compile. Keep compiling the program regularly even if this requires parts of the code to commented out. The order in which the functions are implemented is left entirely up to you. However, this should be done to facilitate incremental development and testing of the program and not necessarily in the order given above.

Submit:

Submit **CardSet.cpp** only for this assignment. Submit using the submit facility on UNIX ie:

$ submit -u *login* **-c CSCI124 -a 3 CardSet.cpp**
where: *login* **is your UNIX login ID and**

We will compile run and print your assignment. The assignment deadline will be strictly enforced. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline must be made to the Subject Coordinator before the submission deadline. Supporting documentation must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the points awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than three days late.