University of Regina
Department of Computer Science
CS210-001 Data Structures and Abstractions (Winter 2016)

Assignment 2 (Assigned: Tuesday, January 26, 2016, Due: Tuesday, February 16, 2016)

Exercises
(**No** programming required, 2 + 3 = 5 Marks)

1. Using **pseudocode** or **C++** code, and the following ADT stack operations, write a procedure

void AppendStack(StackType &S, StackType &T)

to **append**, in **order**, the elements of a stack *T* to the **top** of a stack *S*. For example, if *S* contains the elements "Justin", "Bieber", "sux", in that particular order, from **bottom to top**, and *T* contains the elements "big", "time", in that particular order, from **bottom to top**, then, after the execution of *AppendStack(S, T)*, *S* contains the elements, "Justin", "Bieber", "sux", "big", "time", in that particular order, from **bottom to top**. Note that you may **not** access the internal data members of the stack ADT.

bool *IsEmpty() const*
bool *IsFull() const*
void *Push(ItemType item)*
void *Pop()*
ItemType *Top() const*

2. Using **pseudocode** or **C++** code, and the following ADT queue operations, write a procedure

void RemoveEnd(QueType &Q)

to **remove** the **last** element of a queue *Q*. For example, if *Q* contains the elements 1, 13, 29, 47, in that particular order, then, after the execution of *RemoveEnd(Q)*, *Q* contains the elements 13, 29, 47, in that particular order. Note that you may **not** access the internal data members of the queue ADT.

bool *IsEmpty() const*
bool *IsFull() const*
void *Enqueue(ItemType item)*
void *Dequeue(ItemType &item)*

Programming
(11.5 Marks)

1. Write a program to simulate a **deck of cards**. A card has a **suit**, either *hearts*, *diamonds*, *spades* or *clubs*, and a **value**, in the range of [1,13]. The deck, containing 52 cards, indexed in the range of [0,51], is to be represented by a **stack**, encapsulated in a **class** *CDeck*, with **member functions** to perform the functionally described below.

- A user can **get** the **number** of cards in the deck.
- A user can **add** a card to the **top** of the deck.
- A user can **remove** the **top** card from the deck.
- A user can **shuffle** the cards in the deck. Any method may be used to randomly shuffle the cards in the deck, though you may **not** access the internal data members of the stack ADT.
- A user can **cut** the deck. When cutting a deck, the top $n > 0$ cards of the deck are removed and placed onto a temporary deck *S*, while the other 52 - *n* cards are removed and placed onto a second temporary deck *T*. In both cases, the order of the cards is **not** to be changed as they are moved from the deck to the temporary decks *S* and *T*. The cards of *T* are then **appended** to those of *S*, just as is in the *AppendStack* function of Exercise 1.

Your submission should include a screenshot of the program using the following script.

create a deck
*print all cards in deck, from bottom to top (order of cards **not** to be changed)*
print number of cards in deck

remove top card (as card a)
remove top card (as card b)
remove top card (as card c)
add card a to top
add card c to top
add card b to top
*print all cards in deck, from bottom to top (order of cards **not** to be changed)*
print number of cards in deck

cut deck
*print all cards in deck, from bottom to top (order of cards **not** to be changed)*
print number of cards in deck

shuffle deck
*print all cards in deck, from bottom to top (order of cards **not** to be changed)*
print number of cards in deck