**8.4 Assignment 4, due Thursday 2018-03-01, at 11:00pm**

For this assignment, you are to implement a data type representing a deck of playing cards.

8.4.1 Your task

The file [deck.h](http://www.cs.yale.edu/homes/aspnes/classes/223/examples/2018/hw/4/deck.h), also shown in the [Interface](http://www.cs.yale.edu/homes/aspnes/classes/223/notes.html#hw4_2018_interface) section below, defines the interface to the data type. Your job is to provide a matching deck.c file.

A deck consists of an ordered sequence of 0 or more cards, implemented as the struct card type. Each card has a rank, which is a character in the string "A23456789TJQK", and a suit, which is a character in the string "CDHS". A card is printed by giving the rank and then the suit. For example, the Ten of Diamonds has rank 'T' and suit 'D', and would be printed as TD.

A deck is printed by printing all the cards in the deck, separated by spaces. The deckPrintfunction should do this. There should not be a space after the last card.

The deckCreate and deckDestroy functions create and destroy decks. A new deck always contains 52 cards, ordered by suit, then rank.

The deckGetCard function removes and returns the card at the top of a deck. The deckPutCardfunction adds a new card to the bottom of a deck.

Two additional functions split and combine decks. The deckSplit function takes a deck d, and a number n, and returns (using pointers passed in by the caller) two decks d1 and d2, where d1contains the top n cards in d (or all cards in d if n is greater than or equal to the size of d), and d2contains any cards that are left over. As a side effect, deckSplit destroys d. The deckShufflefunction combines two decks d1 and d2 by alternately taking cards from the top of each deck, starting with d1; if one of the decks runs out, the remaining deck supplies the rest of the cards. Like deckShuffle, deckSplit returns a new deck and destroys its inputs.

A test harness that you can use to try out your code can be found in [testDeck.c](http://www.cs.yale.edu/homes/aspnes/classes/223/examples/2018/hw/4/testDeck.c). This implements *Deck Assembly Language*, a minimalist programming language for manipulating decks. Here is an example of running testDeck by hand. Note that inputs and outputs are interleaved.

$ ./testDeck

# create a new deck and print it

c1 p1

AC 2C 3C 4C 5C 6C 7C 8C 9C TC JC QC KC AD 2D 3D 4D 5D 6D 7D 8D 9D TD JD QD KD AH 2H 3H 4H 5H 6H 7H 8H 9H TH JH QH KH AS 2S 3S 4S 5S 6S 7S 8S 9S TS JS QS KS

# remove top card from the deck

-1 p1

AC

2C 3C 4C 5C 6C 7C 8C 9C TC JC QC KC AD 2D 3D 4D 5D 6D 7D 8D 9D TD JD QD KD AH 2H 3H 4H 5H 6H 7H 8H 9H TH JH QH KH AS 2S 3S 4S 5S 6S 7S 8S 9S TS JS QS KS

# put it back on the bottom

+1 AC p1

2C 3C 4C 5C 6C 7C 8C 9C TC JC QC KC AD 2D 3D 4D 5D 6D 7D 8D 9D TD JD QD KD AH 2H 3H 4H 5H 6H 7H 8H 9H TH JH QH KH AS 2S 3S 4S 5S 6S 7S 8S 9S TS JS QS KS AC

# split into two decks

/ 1 2 17 p1 p2

2C 3C 4C 5C 6C 7C 8C 9C TC JC QC KC AD 2D 3D 4D 5D

6D 7D 8D 9D TD JD QD KD AH 2H 3H 4H 5H 6H 7H 8H 9H TH JH QH KH AS 2S 3S 4S 5S 6S 7S 8S 9S TS JS QS KS AC

# shuffle them back together

\* 1 2 p1

2C 6D 3C 7D 4C 8D 5C 9D 6C TD 7C JD 8C QD 9C KD TC AH JC 2H QC 3H KC 4H AD 5H 2D 6H 3D 7H 4D 8H 5D 9H TH JH QH KH AS 2S 3S 4S 5S 6S 7S 8S 9S TS JS QS KS AC

# split into a nonempty deck and an empty deck

/ 1 2 100000 e1 p1 e2 p2

1

2C 6D 3C 7D 4C 8D 5C 9D 6C TD 7C JD 8C QD 9C KD TC AH JC 2H QC 3H KC 4H AD 5H 2D 6H 3D 7H 4D 8H 5D 9H TH JH QH KH AS 2S 3S 4S 5S 6S 7S 8S 9S TS JS QS KS AC

0

# clean up and exit

d1 d2 q

8.4.2 Interface

*// Abstract data type for a deck of playing cards*

#include <stdio.h>

*// Standard suits and ranks*

#define SUITS "CDHS"

#define RANKS "A23456789TJQK"

*// A single card*

*// This is small enough that we usually pass it*

*// around by copying instead of using pointers.*

**typedef** **struct** card {

char rank; */\* from RANKS \*/*

char suit; */\* from SUITS \*/*

} Card;

*// A deck of cards*

**typedef** **struct** deck Deck;

*// Create a new unshuffled deck of 52 cards,*

*// ordered by suit then rank:*

*//*

*// AC 2C 3C ... KC AD 2D 3D ... KD AH 2H 3H ... KS*

Deck \*deckCreate(void);

*// Free all space used by d.*

*// Running time should be O(length of deck)*

void deckDestroy(Deck \*d);

*// Return true if deck is not empty.*

*// Running time should be O(1).*

int deckNotEmpty(const Deck \*d);

*// Remove and return the top card of a deck.*

*// If deck is empty, behavior is undefined.*

*// Running time should be O(1).*

Card deckGetCard(Deck \*d);

*// Add a card to the bottom of a deck.*

*// This is not required to do anything special if the card is bogus*

*// (e.g. "1C", "?X", "A-").*

*// Running time should be O(1).*

void deckPutCard(Deck \*d, Card c);

*// Split a deck into two piles:*

*// \*d1 is new deck with top n cards in d.*

*// \*d2 is new deck with remaining cards in d.*

*// Order of cards is preserved.*

*// If d contains fewer than n cards, put them all in d1.*

*// Destroys d.*

*// Running time should be O(n).*

void deckSplit(Deck \*d, int n, Deck \*\*d1, Deck \*\*d2);

*// Shuffle two decks together by alternating cards from*

*// d1 and d2 to obtain new deck.*

*//*

*// If d1 is X X X X*

*// and d2 is Y Y Y Y Y Y Y,*

*// return value is X Y X Y X Y X Y Y Y Y.*

*//*

*// If d1 is X X X X*

*// and d2 is Y Y,*

*// return value is X Y X Y X X.*

*//*

*// Running time should be O(length of shorter deck).*

*// Destroys d1 and d2.*

*//*

*// If d1 == d2, behavior of this function is undefined.*

Deck \*deckShuffle(Deck \*d1, Deck \*d2);

*// Print the contents of deck to f as sequence of ranks/suits*

*// separated by spaces.*

*// Example output: "AS TC 9D 3H 5S" (without quotes)*

*// Running time should be O(length of deck).*

void deckPrint(const Deck \*d, FILE \*f);