

```

// ***
// *** You must modify this file
// ***

#include "shuffle.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

// do not modify this function
static void printDeck(CardDeck deck)
{
    int ind;
    for (ind = 0; ind < deck.size; ind ++)
    {
        printf("%c ", deck.cards[ind]);
    }
    printf("\n");
}

#ifdef TEST_DIVIDE
// leftDeck and rightDeck are arrays of CardDeck
// This function creates pairs of left and right decks
// Each pair divides the original deck into two non-overlapping decks
// and
// together they form the original deck.
//
// You can think of the left deck held by the left hand taking some
// cards (at least one) from the top of the original deck.
//
// The right deck is held by the right hand taking some (at least one)
// cards from the bottom of the original deck.
void divide(CardDeck origDeck, CardDeck * leftDeck, CardDeck *
rightDeck)
{
    int size = origDeck.size;
    int j;
    int k;

    for(k = 1; k < size; k++)
    {
        int left = size - k; //number of cards in the left deck
        leftDeck[k - 1].size = left;

        for(j = 0; j < (size - k); j++)
        {
            leftDeck[k - 1].cards[j] = origDeck.cards[j];
        }

        int right = k; //number of cards in the right deck
    }
}

```

```

    rightDeck[k - 1].size = right;

    for(j = (size - k); j < size; j++)
    {
        rightDeck[k - 1].cards[j - size + k] = origDeck.cards[j];
    }
    //printDeck(leftDeck[k]);
    //printDeck(rightDeck[k]);
}
}
#endif

#ifdef TEST_INTERLEAVE

void helper (CardDeck leftDeck, CardDeck rightDeck, CardDeck
shuffledDeck, int leftPos, int rightPos)
{
    int leftSize = leftDeck.size;
    int rightSize = rightDeck.size;

    int currentPos = leftPos + rightPos;

    if(leftPos == leftSize && rightPos == rightSize)
    {
        printDeck(shuffledDeck);
    }
    else
    {
        if(leftPos < leftSize)
        {
            shuffledDeck.cards[currentPos] = leftDeck.cards[leftPos];
            helper(leftDeck, rightDeck, shuffledDeck, (leftPos + 1), rightPos);
        }

        if(rightPos < rightSize)
        {
            shuffledDeck.cards[currentPos] = rightDeck.cards[rightPos];
            helper(leftDeck, rightDeck, shuffledDeck, leftPos, (rightPos + 1));
        }
    }
    return;
}
// Interleave two decks to generate all possible results.
//
// If the leftDeck is {'A'} and the right deck is {'2', '3'}, this
// function prints
// A 2 3
// 2 A 3
// 2 3 A
//

```

```

// If the leftDeck is {'A', '2'} and the right deck is {'3', '4'},
// this
// function prints
// 3 4 A 2
// 3 A 4 2
// A 3 4 2
// 3 A 2 4
// A 3 2 4
// A 2 3 4
//
// Please notice the space does not matter because grading will use
// diff -w
//
// How to generate all possible interleaves?

// Understand that a card at a particular position can come from
// either left or right (two options). The following uses left for
// explanation but it is equally applicable to the right.
//
// After taking one card from the left deck, the left deck has one
// fewer card. Now, the problem is the same as the earlier problem
// (thus, this problem can be solved by using recursion), except one
// left card has been taken. Again, the next card can come from left
// or right.
//
// This process continues until either the left deck or the right deck
// runs out of cards. The remaining cards are added to the result.
//
// It is very likely that you want to create a "helper" function that
// can keep track of some more arguments. If you create a helper
// function, please keep it inside #ifdef TEST_INTERLEAVE and #endif
// so that the function can be removed for grading other parts of the
// program.
void interleave(CardDeck leftDeck, CardDeck rightDeck)
{
    int leftPos = 0;
    int rightPos = 0;

    CardDeck shuffledDeck =
    {
        .size = (leftDeck.size + rightDeck.size),
        .cards = {0}
    };

    helper(leftDeck, rightDeck, shuffledDeck, leftPos, rightPos);
}
#endif

#ifdef TEST_SHUFFLE
// The shuffle function has the following steps:

```

```

// 1. calculate the number of possible left and right decks. It is
// the number of cards - 1 because the left deck may have 1, 2,...,
// #cards - 1 cards.
//
// 2. allocate memory to store these possible pairs of left and right
// decks.
//
// 3. send each pair to the interleave function
//
// 4. release allocated memory
//
void shuffle(CardDeck origDeck)
{
    int numDecks = origDeck.size - 1; //number of possible divisions

    CardDeck * leftDeck = malloc(sizeof(CardDeck) * numDecks); //
    allocates memory for leftDecks
    CardDeck * rightDeck = malloc(sizeof(CardDeck) * numDecks); //
    allocates memory for rightDecks

    divide(origDeck, leftDeck, rightDeck);

    for(int i = 0; i < numDecks; i++)
    {
        //printDeck(leftDeck[i]);
        //printDeck(rightDeck[i]);
        interleave(leftDeck[i], rightDeck[i]);
    }

    free(leftDeck);
    free(rightDeck);
}
#endif

```