Lab 10: Structures and File I/O...

Due date: Friday, December 5, 11:59pm.

Assignment Preparation

Lab type. This is a pairs lab. Select your partner at will. For this lab, you get to select your own partner.

Collaboration. Students work in pairs, and it is considered cheating, if members of the team (pair) do not work together. Communication between pairs during lab time is allowed, but no direct sharing of code is allowed.

Purpose. Continuation of work with C structures and file I/O.

Programming Style. All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/∼cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

Testing and Submissions. Any submission that does not compile using the

```
gcc -ansi -Wall -Werror -lm
```

compiler settings will receive an automatic score of 0.

Assignment: poker.c

Write a program that reads a description of a poker hand from a file and analyzes it. Name your program `poker.c`. Note, the program will contain multiple functions, but you should put all of them into a single file, without creating the .h file.

Card deck. Your program will work with a standard poker card deck. The card deck consists of four suits, clubs, hearts, diamonds and spades. Each suit contains 13 cards, with the following denominations: 2,3,4,5,6,7,8,9,10, Jack, Queen, King and Ace. Each card, therefore, can be described by a pair of values: its suit and its denomination, or value.

Game of poker. We consider the traditional variant of the game of poker. In it, a player is dealt five cards. The player then has an option of replace any of the cards from his/her hand. The goal of each player in the game is to assemble the best possible combination of cards. In our program, we will be interested in the following combinations:

1. Flush. All five cards in the player's hand belong to the same suite.
2. Four of a kind. The player's hand contains four cards with exactly the same value.

Program flow. Your program will take its input from a file called poker-input.txt. The input will consist of a description of an initial poker hand, instructions on which cards need to be traded, and the descriptions of new cards. Your program shall output the poker hand, check if it contains a flush or a four-of-a-kind combination, then perform the replacement of the cards in the hand, and perform the flush and four-of-a-kind checks again.

Data structures. Your program shall use two struct types to describe individual cards in the deck/hand, and to describe a poker hand. Define your structures as follows:

```
#define HAND_SIZE 5
typedef struct {
     /* your structure definition goes here */
     /* ... */
} cardType;

typedef struct {
    cardType cards[HAND_SIZE];
     /* add any other information about the hand you deem necessary */
     /* ... */
} handType;
```

Input specification. The input to your program will be a text files with the name poker-input.txt. The file will contain the following information:

1. Initial hand. The initial hand dealt to a player consists of five cards. For each card, two values are specified: the suite and the denomination.

   The suite will be specified as a single character. Possible values are:

'C': clubs
'D': diamonds
'H': hearts
'S': spades

The `denomination` of the card will be specified as one of the following values: `0, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A`. Note that some of these values appear to be numbers and some — characters. Note also, that one value, `10`, contains two digits/characters, while all other values contain only one. **Your code will have to deal with this properly**.

Each card description will occupy a single line of the input file and will first list the value and then, the suite of the card. E.g., the following input:

```
10 C
J H
K H
7 S
2 D
```

specifies the following hand: 10 of clubs, Jack and King of hearts, 7 of spades and 2 of diamonds.

2. Number of exchanges. An integer number between 0 and 5. This number will occupy a single line of the input file. It specifies how many cards are to be exchanged. 0 means, no cards are exchanged, 1 means – one card, etc.., 5 means that all five cards need to be exchanged.

3. Cards to be exchanged. The next line of input will contain 0 — 5 integer numbers. Each number will be between 1 and 5 and no two numbers will be the same. These numbers specify which cards need to be replaced in the hand. The number of integers in the line is equal to the number of exchanges specified in the line above.

   Consider, for example, the following input:

```
10 C
J H
K H
7 S
2 D
2
4 5
```

   The initial hand is as described above. The sixth line of the input file specifies that the player wants to exchange two (2) cards. The seventh line specifies that these are the fourth and the fifth cards in his/her hand, i.e. the 7 of spades and the 2 of diamonds.

4. Replacement cards. The last lines of the input file will contain replacement cards. The number of replacement cards shall be equal to the number of cards to be exchanged. The format for each replacement card is the same as the format for each card in the original hand: value of the card goes first, followed by the suite.

Example. Consider the following *complete* input file:

```
10 C
J H
K H
7 S
2 D
2
4 5
6 H
A D
```

This input file describes the following. The starting hand for the player is *10 of clubs, Jack and King of hearts, 7 of spades and 2 of diamonds.* The player makes decision to exchange two cards: *7 of spades* and *2 of diamonds.* These cards are replaced with the *6 of hearts* and *ace of diamonds.*

Note: It is possible to have 0 (zero) cards replaced. In this case the input file will contain no lines after the line specifying the number of cards to be exchanged. For example, the following *complete* input file specifies the same initial hand as above with no exchanges:

```
10 C
J H
K H
7 S
2 D
0
```

Output specifications. Your program shall provide the following output:

1. description of the initial hand together with its analysis.
2. description of the exchanges.
3. description of the final hand together with its analysis.

Description of the initial hand. Your program shall output full name of each card in the hand. The full name consists of the full value of the card (2,3,4,5,6,7,8,9,10, `"Jack"`, `"Queen"`, `"King"` `"Ace"`) followed by the word `"of"` followed by the name of the card suite: `"Clubs"`, `"Diamonds"`, `"Hearts"` or `"Spades"`. Each card is prefaced with its number in the hand and is printed out on a separate line. For example, given the input from above, the following text shall be printed to describe the initial hand:

```
Initial hand:
  1. 10 of Clubs
  2. Jack of Hearts
  3. King of Hearts
  4. 7 of Spades
  5. 2 of Diamonds
```

Analysis of hand. Your program shall check (see below) if a hand contains a flush or a for-of-a-kind combination. It shall report the findings after outputing each hand (both the initial and the final). The output shall look as follows:

1. If a hand has a flush, the following shall be printed to the standard output:

   `You have a flush of <SUITE>`

   where `"<SUITE>"` is one of `"Clubs"`, `"Diamonds"`, `"Hearts"`, `"Spades"`.

2. If a hand has a four-of-a-kind combination, the following shall be printed to the standard output:

   `You have four <Value>s`

   Here, value is one of 2,3,4,5,6,7,8,9,10, `"Jack"`, `"Queen"`, `"King"`, `"Ace"`.

3. If a hand has neither flush, no four-of-a-kind, your program shall print to the standard output the following text:

   `Better luck next time!`

Example. For the following input:

```
10 C
J H
K H
7 S
2 D
```

the analysis will output

```
Better luck next time!
```

For the input hand:

```
10 C
J H
J D
J S
J C
```

the analysis will output

```
You have four Jacks
```

For the input hand:

```
10 C
5 C
8 C
A C
J C
```

the analysis will output

```
You have a flush of Clubs
```

Description of transition. Your program shall specify how many cards are being changed. Thenm for each card to be exchange, your program shall display the full description of its card, and the card it is replaced with. The description of each card is the same as above.

The exact output format is shown in the example below.

Example. Consider the following *complete input*.

```
10 C
J H
K H
7 S
2 D
2
4 5
6 H
A D
```

The description of transitions, printed by the program shall look as follows:

```
Changing 2 cards:
     7 of Spades ---> 6 of Hearts
     2 of Diamonds ---> Ace of Diamonds
```

Description and analysis of final hand. After the replacement procedure has been described, the program shall display the final hand, and analyze it. The output format for the final hand and the output format for its analysis are *exactly the same* as the output format for the initial hand and for the results of its analysis.

Example. For the input file above, the complete output will be:

```
Initial hand:
  1. 10 of Clubs
  2. Jack of Hearts
  3. King of Hearts
  4. 7 of Spades
  5. 2 of Diamonds

  Better luck next time!

Changing 2 cards:
     7 of Spades ---> 6 of Hearts
     2 of Diamonds ---> Ace of Diamonds

Final hand:
  1. 10 of Clubs
  2. Jack of Hearts
  3. King of Hearts
  4. 6 of Hearts
  5. Ace of Diamonds

  Better luck next time!
```

For the following input:

```
J H
J C
7 C
8 S
J D
2
3 4
J S
5 S
```

the output will look as follows:

```
Initial hand:
  1. Jack of Hearts
  2. Jack of Clubs
  3. 7 of Clubs
  4. 8 of Spades
  5. Jack of Diamonds

  Better luck next time!

Changing 2 cards:
     7 of Clubs ---> Jack of Spades
     8 of Spades ---> 5 of Spades

Final hand:
  1. Jack of Hearts
  2. Jack of Clubs
  3. Jack of Spades
  4. 5 of Spades
  5. Jack of Diamonds

  You have four Jacks
```

For the input

```
J H
J C
7 C
8 S
J D
0
```

the output will be

```
Initial hand:
  1. 10 of Clubs
  2. Jack of Hearts
  3. King of Hearts
  4. 7 of Spades
  5. 2 of Diamonds

  Better luck next time!
```

```
Changing 0 cards:

Final hand:
  1. 10 of Clubs
  2. Jack of Hearts
  3. King of Hearts
  4. 7 of Spades
  5. 2 of Diamonds

  Better luck next time!
```

Error handling. You are responsible for the following potential input errors:

1. Invalid card. One of the cards specified in the input file has either incorrect suite or denomination. Examples of invalid cards are:

   ```
   11 S
   Jack H
   -4 H
   5 Hearts
   A Diamonds
   Ace Diamonds
   ```

   If this error is encountered, output

   ```
   Error: incorrect card: <Value> <Suite>
   ```

   where `<Value>` and `<Suite>` are the values read from the input file.

2. Incorrect number of cards to be replaced. The number of cards to be replaced shall range from 0 to 5 (inclusively). Any other number is invalid. Your program shall output:

   ```
    Error: invalid number of cards to exchange: <Number>
   ```

   where `<Number>` is the (invalid) number of cards to be exchanged read from the input file.

3. Ivalid cards to be replaced. There are five cards in a hand, they are numbered in the order in which they are read from the input file. In the line specifying which cards are to be replaced each number shall be between 1 and 5. If an invalid number is detected, your program shall output

   ```
     Error: invalid card number: <Number>
   ```

   where `<Number>` is the (invalid) card index number read from the input file.

If any of these errors are encountered, your program shall stop working after the error message is printed.

Your program can assume that the following will hold:

- The number of integers in the seventh input line will be equal to the value that appears in the sixth input line.
- The number of lines below the seventh input line will be equal to the value that appears in the sixth input line.
- There are no duplicate cards in the input (neither in the initial hand, nor among the replacement cards).

All input files to your program will possess the properties above.

Functions.

Your program shall contain the following functions:

```
cardType readCard(FILE *f);
int isFlush(handType hand);
int isFour(handType hand);
handType replaceCard(handType hand, int index, cardType card);
void toString(char str[], cardType card);
```

cardType readCard(FILE *f). This function reads from the input `FILE f` the denomination and the suite of a card, and prepare and output a variable of type `cardType` describing the the card. (This is where you handle input format issues).

int isFlush(handType hand). This function takes as input a variable `hand` describing a poker hand (of type `handType`). It outputs `1` if the `hand` contains a flush, i.e., if all cards in poker hand are of the same suite. It outputs `0` otherwise.

int isFour(handType hand). This function takes as input a variable `hand` describing a poker hand (of type `handType`). It outputs `1` if the `hand` contains a for-of-a-kind combination, i.e., four cards in the hand have the same value. It outputs `0` otherwise.

handType replaceCard(handType hand, int index, cardType card). This program takes as input a variable describing a poker hand, a variable describing a single card, and the index of the card in the hand to be replaced. The output of the function is a poker hand where `card` replaces the card at position `index`.

Note: Be consistent in what values `index` takes. You can either make it take values from 1 to 5 (directly from the seventh input line) or you can make it take values from 0 to 4 (corresponding directly to array indexes). Either decision is fine as long as you correctly handle the value of the `index` variable in your code.

void cardDesc(char str[], cardType card). This function takes as input a description of a card. It produces, in its out parameters `str` the string description of the card, which will be used by your program to print the output.

Submission procedure.

You need to submit two files:`team.txt` and `poker.c`. Put the names and the email addresses of both partners in the `team.txt` file.

You will be using `handin` program to submit your work. The procedure is as follows:

- `ssh` to `vogon` (`vogon.csc.calpoly.edu`).
- Students from **Section 009** shall execute the following submission command:

```
> handin dekhtyar-grader lab10-09  <your files go here>
```

- Students from **Section 011** shall execute the command:

```
> handin dekhtyar-grader lab10-11 <your files go here>
```

`handin` is set to stop accepting submissions 24 hours after the due time.