Programming Assignment 9

Arrays and structures

Purpose of the exercise

This exercise will help you do the following:

1. Develop skills in C programming with arrays and struct definitions.
2. Apply your skills to implement a solitaire card game.

Overview

Accordion is a solitaire game for one player using a single deck of playing cards. It is so named because it looks like accordion pleats which have to be ironed out. The rules of the game are as follows:

> A standard playing deck is used. The cards from the entire deck are spread out in a single line. Whenever the card matches its immediate neighbor on the left, or matches the third card to the left, it may be moved onto that card. Cards match if they are of the same suit or same rank. After making a move, look to see if it has made additional moves possible. When a card is moved, all the cards in the pile are moved. Gaps left behind are filled by moving all piles on the right of the gap to the left. Deal out the whole pack, combining cards towards the left whenever possible. The game is won if the pack is reduced to a single pile.

In this assignment, you are to implement functions that can play the game. For this exercise note the following: (1) Where there is a choice between making a three-position move or a one-position move, choose a three-position move. (2) Where there is more than one card that may be moved three positions, choose the leftmost card. (3) If no three-position move is possible and more then one one-position move may be made, select the leftmost move. Here's an example:

`H3` (Heart ♥3) `C2` (Club ♠2) `DT` (Diamond ♦10) `HT` (Heart ♥10) `CK` (Club ♣King)

According to the sequence of the cards and the above guidelines, `HT` (Heart ♥10) should be placed over `H3` (Heart ♥3), because `H3` is the third matching card to the left of `HT`, and `HT` is the leftmost card that may be moved three places.

The input to the program specifies the cards for several games; each line of input contains the cards for one game. Each line may contain from 1 to 52 cards separated by a single space; Cards are represented as a two character code:

- The first character is the suit
  - **C**lub ♣
  - **D**iamond ♦
  - **H**eart ♥
  - **S**pade ♠
- The second character is the rank
  - **1** for Ace,
  - **2**...**9**,
  - **T** for **10**,
  - **J**ack,
  - **Q**ueen,

- King.

The line ends with two characters `00` (double zeroes); consider it a special card.

The output of the program shows the cards remaining after playing a round of the game with the pack of cards as described by the corresponding input line. A line of output must be produced for a line of input.

The input of the above example is:

```
H3 C2 DT HT CK 00
```

The output line is:

```
HT C2 DT CK 00
```

Task

1. Download the assignment source code files:

Go to the course page in *Moodle* - DigiPen (Singapore) online learning management system, download a zipped archive that contains all the source code files and extract them into a Microsoft Windows directory *c:\sandbox\* (adjust the path as suitable for your system). The included files are:

- *qdriver.c*
- *gamelib.c*
- *gamelib.h*
- *cardlib.h*
- *input.txt*
- *expected-output.txt*

2. Using the Microsoft Windows command prompt navigate to the sandbox folder. Then type `wsl` to open Linux bash in the same current directory.
3. From Linux open *gamelib.h* in Microsoft Visual Studio Code.

Study the function declarations in this file. Take note that the `main()` function from *qdriver.c* reads the game line-by-line from the standard input, stores each game into an array of characters. Your task is to define functions for loading cards from that array into an array of cards, playing the game, and displaying results:

```c
// Loads a game by reading it from text[] into the array of cards.
void load_game(const char text[], Card game[]);
// Plays a game.
void play_game(Card game[]);
// Displays the sequence of cards.
void display_game(const Card game[]);
```

4. Each `game` is an array of `Card` objects. This and other definitions can be found in *cardlib.h*:

```c
typedef char CardSuit;
typedef char CardRank;

typedef struct Card
{
    CardSuit suit;
    CardRank rank;
} Card;
```

Get familiar with the contents of this file.

5. Inside *gamelib.c* you will find a skeleton of the code that you have to complete. You will find there one more function you must define. A helper function `del_card()` is also required in the program; it takes a `game` and an index representing a position of a `Card` object to delete from the game; deleting means shifting all cards after the indicated `position` one place to the left to fill the gap caused by the deletion.
6. Complete the definitions of all four required functions inside *gamelib.c*.
7. Compile the program:

```
gcc -Wall -Wextra -Wconversion -Wstrict-prototypes -Werror -pedantic-errors -std=c11 -o main gamelib.c qdriver.c
```

8. Run executable main:

```
./main < input.txt > actual-output.txt
```

9. Use diff to compare the actual output with the expected output:

```
diff expected-output.txt actual-output.txt
```

Make sure that the output matches exactly.

Submitting the deliverables

You have to upload a complete file *gamelib.c* to Moodle - DigiPen (Singapore) online learning management system where the file will be automatically evaluated.