ECE 175: Computer Programming for Engineering Applications

Final Project: *Go Fish*

Due Date: Monday December 3rd, 2018, 2:00 PM via D2L Dropbox

1 Administrative Details

1.1 Groups

You are to work in groups of **strictly** two students. Team members will be individually questioned on the entire code. Team members need to clearly articulate the responsibilities of each member and demonstrate knowledge of the entire code.

1.2 Scheduling Project Demo

You have to appear as a group to demonstrate your code.

Sign up your team for the final project demo here: https://tinyurl.com/ycghmkpb

Sign up for the demo here: https://tinyurl.com/yaatc6ea

Slots are available on a first-come-first-serve basis. Teams or team members that do not demo their code will receive zero credit for the final project.

1.3 Intermediate Code Design Submission

To keep track of your progress, an intermediate code submission is required. The deadline for the intermediate code submission is on November 28, 2018 @ 2:00 PM. Your intermediate submission will be checked on that week's lab. Please submit:

1. A high-level design of your code in a PDF. You are free to use any format that makes sense to your team.
2. An initial code implementation, with a deck of cards being implemented as a linked list and a shuffling function that shuffles the deck.
3. A skeleton of the remaining of the code containing all the function prototypes and comments on the input/output of the functions and their intent.

1.4 Final Code Submission

The project is due on December 3rd @ 2:00 PM. Submit your code via D2L – one submission from a group member is sufficient. You may also submit any design documents that will help you explain your code.

Late submission policy: 10% deduction per day. The last date of final project demo is by 5PM on Wednesday December 6, 2017. After this time the project will NOT be accepted/graded.

1.5 Academic Integrity Policy

Each group is expected to submit its own code. You may ask other for advice, and in general discuss the project, but you should WRITE YOUR OWN CODE. If any part of the code submitted by different students is identical, ALL involved parties will receive zero credit on the entire project and one letter reduction in their final grade. This policy will be very aggressively enforced. ALL submitted code will be checked with a plagiarism detection tool (e.g., http://theory.stanford.edu/~aiken/moss/).

1.6 Useful Suggestions

a) Spend time designing your code and use modular programming. Create all the function prototypes and describe what they do before developing your code. Create pseudocode of your program flow.
b) Determine test cases for your functions and your overall code to ensure proper functionality.
c) Write well-documented code to ensure you can explain it during demo.

2 A Game of Go Fish

You are to develop an interactive game of Go Fish between two players. In Go Fish, each player takes turns asking for a card with a specific rank or face from another player. The goal is to form a book (a set of 4 cards with the same face or rank). The player with the most books at the end of the game is declared the winner. The gameplay for Go Fish is described at https://en.wikipedia.org/wiki/Go_Fish. Your program should operate as follows.

2.1 Setup

1. Go Fish cards are represented as variables of the following type:

```
typedef struct card_s {
        char suit[7];
        int value;
     struct card_s *pt;
} card;
```

You are allowed to add attributes to this definition, but not to remove any. You can represent colors by using card suits. Red: hearts; Yellow: diamonds; Green: clubs; Blue: spades. The action field is used to denote the function of action cards.

2. The game is played with a standard 52-card deck.
3. At the beginning, the user can choose to shuffle the deck or load a predefined sequence of cards from a file (for testing). The cards are stored in a file in a format of your choice. For instance, they could be stored in the form

   8 diamonds
   9 hearts
   10 spades
   ….

4. The deck is implemented by a dynamic list of cards. This can be a singly-linked or a doubly-linked list (recommended). The cards drawn from the deck are deleted from the list.
5. Each player's hand is implemented by a dynamic list of cards. The list is initially populated with the cards dealt to each player. The card drawn (played) by each player is added to (deleted from) the respective list.
6. The pool of cards in the center is implemented as a dynamic list of cards that is shuffled at the beginning of your game. The cards drawn or discarded in the pile is added/deleted in a dynamic fashion.

2.2 Game Play

The gameplay is described at https://en.wikipedia.org/wiki/Go_Fish

2.3 Optional Features for Extra Credit

1. **Any number of players:** Modify your code to work with more than two players. Look at the rule changes when more than 4 players participate (in terms of card dealing).
2. **Game variations:** Implement at least four game variations, as they are described in the game description.
3. **Player automation:** Automate one of the players to implement to play the game according to a winning strategy
4. **Graphics.** Add graphics to your game. You can print cards using ascii art. For example, you can make cards appear as

```
----------      ----------
|9♥     |      |K♣     |
|       |      |       |
|       |      |       |
|    9♥ |      |    K♣ |
----------      ---------
```

2.4 Sample Execution

Note: You are encouraged to make your output look more appealing than what is given below. Your interface should display sufficient direction for a user to play the game. Get friends and family to play your game without much assistance.

It is very likely that you will not get the exact same code execution shown below. This is used to show you how the game should be played.

Enter your name player 1: Wilma
Enter your name Player 2: Alpha

```
|---------  ---------  ---------  ---------  |
|---------  ---------  ---------  ---------  |
          Wilma, Alpha
        Let's play Go Fish
|---------  ---------  ---------  ---------  |
|---------  ---------  ---------  ---------  |
```

Wilma's hand has 7 cards
A♠ A♥ 2♠ 2♥ 3♠ 3♥ 4♠

Alpha's hand has 7 cards (debugging purpose)
A♦ A♣ 2♦ 2♣ 3♦ 3♣ 4♦

*** Hit C to Continue ***: C

Wilma, your turn

Wilma's hand has 7 cards
A♠ A♥ 2♠ 2♥ 3♠ 3♥ 4♠

Wilma, pick a card (2 - 10, J, Q, K, A): Y
This is not a valid card.

Wilma, pick a card (2 - 10, J, Q, K, A): 4

Wilma, card 4♦ is in your hand.

Taking card(s) from Alpha's hand

*** Hit C to Continue ***: c

Wilma, your turn

Wilma's hand has 8 cards
A♠ A♥ 2♠ 2♥ 3♠ 3♥ 4♠ 4♦

Wilma, pick a card (2 - 10, J, Q, K, A): 4

><(((('> ><(((('> ><(((('>
><(((('> ><(((('>
~~~ Go Fish ~~~
><(((('> ><(((('>
><(((('> ><(((('> ><(((('>

Wilma, take card from the pool

If the card has the same rank that you asked for, you get another turn.

Wilma's hand has 9 cards
A♠ A♥ 2♠ 2♥ 3♠ 3♥ 4♠ 4♦
4♥

*** Hit C to Continue ***: C
~~~

ECE175
Grading Rubric for Final Project

Group Names: Grading time:

	Criterion	Maximum Points	Exemplary (100%)	Proficient (75%)	Marginal (25%)	Unacceptable (0%)
	Requirements					
	Intermediate Grading					
1	Deck creation	/5	Deck is implemented as a linked list. The list is properly initialized and can be traversed. All cards are present in the deck	Deck is implemented as a linked list. However the implementation is incomplete, not all cards are present, or the list is not probably traversed	Some skeleton of a list is available, but the code does not properly compile	The deck is not implemented as a list.
2	Deck Printing	/5	Deck is correctly printed from the list	Deck is printed but the card format does not appear properly, making difficult to play the game	The print function does not work properly but some skeleton of the function is present	No printing function implented.
3	Deck suffling	/5	Deck is properly shuffled to a random deck	Deck is partially shuffled, but the deck does not appear sufficiently random	Deck shuffling breaks upon execution	Deck is not shuffled
4	Pseudo-code and code skeleton	/5	A pseudo-code is presented that accounts for all the game mechanics. Code is sufficiently modular and function prototypes are prototyped	A pseudo-code is presented, but several game functionalities are not accounted for. The code is not modular and only a few functions are prototyped	The pseudo-code is scarce and lacks essential details. Hadly any functions are prototyped.	No pseudo-code or functions are defined.
	Final Project Grading					
1	Deck creation	/5	Deck is implemented as a linked list. The list is properly initialized and can be traversed. All cards are present in the deck	Deck is implemented as a linked list. However the implementation is incomplete, not all cards are present, or the list is not probably traversed	Some skeleton of a list is available, but the code does not properly compile	The deck is not implemented as a list.
2	Deck suffling	/5	Deck is properly shuffled to a random deck	Deck is partially shuffled, but the deck does not appear sufficiently random	Deck shuffling breaks upon execution	Deck is not shuffled
3	Card Dealing	/5	Cards are dealt in the proper order.	The right number of cards is dealt, but not in a proper order	Cards are not dealt correctly	Card dealing is not implemented
4	Players' hands	/5	Players hands are impemented as a linked list. Memory is dynamically allocated with every card draw and freed after one card is played.	Players' hands are implemented as a dynamic list, but memory is not properly handled	Players hands are not implemented as a dynamic list	Players hands are not implemented
5	The Pool	/5	The pool is impemented as a linked list. Memory is dynamically freed as cards as drawn from the pool.	The pool is implemented as a dynamic list, but memory is not properly handled	The pool is not implemented as a dynamic list	A pool is not implemented
7	Game rules	/20	All game rules are followed	Most game rules are followed	Most game rules are not followed	Game is not functional
8	Game Interface	/10	The interface is inuitive. The user is able to play the game with minimal isntructions. Transitions from round to round are clear	The interface is intuitive for the most part. Some difficulty in understanding the game navigation.	The interface is counter-intuitive. Navigation options are not clearly stated. Interface limitations prevent proper game functionality	The interface is very basic and does not allow transitions between game rounds.
	Program Design					
9	Code modularity	/10	The code is logically divided to several functions that implement important functionality (deck shuffling, dealer hand, player hand, hand tallying, list creation, list deletion, etc.)	The code is modular but further simplification could have been attempted	The code only has a few functions. Most functionalities are integrated within the main function	Code is not modular
10	Code documentation	/5	The code is properly documented. The input/output and goal of every fucntion is adequately described. Comments are provided for various parts of the code	The code is partially documented	The code is only cursorily documented	No documentation is provided

11	Compilation	/10	Code succesfully compiles without errors or warnings. The code does not hang while in execution	The code succesfully compiles, but some conditions may make it hang	N/A	Code does not compile
Total		**/100**				

Extra Credit						
12	Player Automation	/5	One of the players is played by the computer			
13	Multiple Players	/5	User can choose between 2 and 10 players			
14	Game Variations	/5	The progressive Uno and Seven-O rules are implemented			
15	Graphics	/5	Ascii art is used to display cards			
Total		**/20**				