

**CS215: Introduction to Program Design, Abstraction and Problem Solving
(Fall, 2021)**

**Programming Assignment 3
(100 points)**

Today's Date: Tuesday, November 16

Due Date: December 5

Problem Statement

Write a program that plays a simple card game, named War (also known as Battle in the United Kingdom) ([https://en.wikipedia.org/wiki/War_\(card_game\)](https://en.wikipedia.org/wiki/War_(card_game))), typically played by two players using a standard playing card deck. The objective of the game is to win all of the cards, and often played by children. The game is played as follows:

1. Each player gets dealt half the deck, 26 cards, and the cards are put face down in the pile in front of the players.
2. Both player turn their top card face up at the same time. The person with the higher card wins the draw, and takes both the cards. They are put to the bottom of the pile, which the player can continue using cards on his/her pile. Aces are high, and suits are ignored.
3. If the two cards played are of equal value, then there is a "war". Both players place the next three cards face down and then another card face-up. The owner of the higher face-up card wins the "war" and adds all the cards on the table to the bottom of the winner's pile. If the face-up cards are again equal, then the battle repeats with another set of face-down/up cards. This repeats until one player's face-up card is higher than his/her opponent's or one player does not have enough cards to finish the war then loses immediately.
4. First player to finish all his/her cards loses the game.
5. If a player finishes his/her cards during a "war" without having enough cards to finish the "war" then loses immediately.

Part 1: Design your own Player class for War Game

For this project, you will be given the complete definition of two classes: **Card** and **Deck**, which you can directly use for your project. The following shows the work you need to do during Lab 11 class:

Download a zip file named `Lab11.zip` from the following link to your computer: (<http://www.cs.uky.edu/~yipike/CS215/Lab11.zip>) and choose to "Extra All" when you right click the zip file.

Then, double click the file named `Lab11.sn`, and it should open the solution, which contains **FIVE** source files: `card.h`, `deck.h`, `card.cpp`, `deck.cpp` and `Lab11.cpp`. You can compile and test running the program to understand the purpose of the program. Test running this program in two different cases: (1) without changing anything in the original solution you downloaded; (2) activate the statement at line number 28 of `Lab11.cpp` by removing the comments sign `//`, then compile and run the program again. What are the different outputs from above two testing cases? Why? Get familiar with the

definitions of two classes and how to use their member functions so that you can use them for your Project 3. For example, how to represent a 52-card deck? How to print a card? How to store the suit and the point of a card? How to create a 52-card deck without generating two cards of the same suit and point value, which is not allowed in the real card game? And so on.

After you get familiar with the definitions of classes named **Card** and **Deck**, you can start to design the class named **Player**, which represents the pile of cards in one player's hand and the actions that a player may take during the War game, such as **play_a_card**; **addCards** when a player wins a round and gets all the cards on the table; **dropCards** when there is a tie, each play needs to drop 3 cards (face down) on the table, then play one more card (face up); and so on. The following shows an incomplete design of this class:

```
class Player
{
    public:
        // default constructor
        Player();

        // alternative constructor
        Player(vector<Card> ini_cards);

        // return how many cards player holds currently
        int getNumCards() const;

        // player plays one card at the front of cards at hand
        Card play_a_card();

        // player wins and adds winning cards to the end of the pile at hand
        void addCards(vector<Card> winningCards);

        // player drops THREE cards from the front of pile at hand
        // when there is a tie
        vector<Card> dropCards();

        // display cards at player's hand
        void print() const;

        // you are allowed to add other member functions if you want

    private:
        int numCards;           // how many cards in player's hand
        ???<Card> cards;       // sequence of cards in player's hand
};
```

In Lab11, you need to complete the declaration of this class in the file named **Player.h**. You need to **exactly match** the design highlighted **in blue** from the above class declaration, and create your own design for the private data member highlighted in red. The private data member named **cards**, represents the pile of cards in player's hand. Which data structure should you choose to store the sequence of cards so that it can

support “removing cards from one side and adding cards from another side of the pile efficiently”. Till now we have introduced data structures such as arrays, vectors, lists, stacks and queues. Make your own choice of data structure to replace “???” highlighted in red and explain to your TA during Lab 11 class why you make such choice. Try to finish the complete definition of the class named **Player** during Lab11 class and test your definition using the main function in **Lab11_testPlayer.cpp**.

Part 2: Complete the definition of Player class and provide main function for War Game

You can either make a copy of Lab11.zip and change the file name of Lab11.cpp into Project3.cpp or create a new empty project, named Project3, then copy and add all source files you need to Project3 solution.

Provide the complete definition of class named **Player**, if you have not finished it during Lab11 class.

Start to write the main function to demonstrate the War game between two players:

1. Display one top card (suit and point) from each player, which represents the card played by each player in the current round
2. Display how many cards on the pile (on the table)
3. Decide which player wins the current round or it is a tie
 - If one player wins, display “Player x wins...get all cards from the pile!”
 - If it is a tie, display “Each player drops three cards (face down) on the pile, then play one more card (face up)”
4. Display how many cards in player1’s hand and how many cards in player2’s hand
5. After each round, your program should ask the user “Do you want to continue...for the next round? (N or n to quit the game).
 - If the user clicks enter key, the game should continue to the next round, back to step 1
 - If the user clicks either “N” or “n” to stop the game, your program should display the following information, then quit. “You choose to quit the game! Player1 has XXX cards left! Player2 has YYY cards left!” where XXX and YYY are the number of cards in each player’s hand at that moment respectively.
6. First player to finish all his/her cards loses the game, and your program should stop and report who wins the game.
7. Your program should also stop immediately if one player finishes his/her cards during a “war” without having enough cards to finish the “war”, then report who wins the game.

8. If both player finish cards at the same time, your program should report a tie game then stop.

Please download the following sample output file to test running your program, and especially check THREE testing cases described in the following pdf file:

<http://www.cs.uky.edu/~yipike/CS215/PA3Sample.pdf>

If you can demonstrate your Project 3 during Lab12 class, you may gain at maximum 5 bonus points for Lab12.

Submission:

Open the link to course Canvas page (<https://www.uky.edu/canvas/>), and log in to your account using your linkblue user id and password. Please submit THREE files (Player.h, Player.cpp and Project3.cpp) through the submission link for “**Project 3**”. It is your responsibility to check whether your submission is successful. If it is not, submit it again till you get the confirmation that your submission is successful.

(Late assignment will be reduced 10% for each day that is late. The assignment will not be graded (you will receive zero) if it is more than 3 days late. Note that a weekend counts just as regular days. For example, if an assignment is due Friday and is turned in Monday, it is 3 days late.)

Always read the grading sheet for each project assignment. It lists typical errors. Check for these errors before submitting your source code. **Please note that your C++ program must compile in order to be graded. If your program cannot pass the compilation, you will get 0 point.**

(The grading sheet is on the next page.)

Grading Sheet for Project Assignment 3

Total: 100 points.

These are example errors. There are other ways to lose points. C++ programs must compile in order to be graded	Points	Deducted Points
Correctness	60	
Provide the correct main function to follow the description of the War game in the problem statement. Your program repeatedly doing the following until the game is over either by the user or one player runs out of card:		
*Correctly display one top card (suit and point) from each player, which represents the card played by each player in the current round	5	
*Correctly display how many cards on the pile (on the table)	5	
*Correctly decide which player wins the current round or it is a tie	5	
*Correctly display how many cards in player1's hand and how many cards in player2's hand	5	
*After each round, your program should ask if the user want to continue and take actions accordingly	5	
*First player to finish all his/her cards loses the game, and your program should stop and report who wins the game.	5	
* Your program should also stop immediately if one player finishes his/her cards during a "war" without having enough cards to finish the "war", then report who wins the game.	5	
Provide the correct declaration of Player class (make your own choice of the data structure for the private data member named cards in Player.h	5	
Provide the correct implementation of member functions for Player class in Player.cpp	18	
Provide separate .cpp file and header file for class named Player	2	
Style	10	
Lay out your program in a readable fashion	3	
Include comments as specified in the lecture notes	4	
User-friendliness in I/O design	3	
Testing (No Documentation is required)	30	
Pass testing case 1 described in Sample output pdf file: exactly match the sample output under testing case 1;	10	
Pass testing case 2 described in Sample output pdf file: the user chooses to quit the program before the game is over and correctly report how many cards in each player's hand;	10	
Pass testing case 3 described in Sample output pdf file: your program needs to continue playing without the interaction with the user, and correctly decide which player wins the game or it is a tie game.	10	
Your Score		