FULL SAIL UNIVERSITY

COP2334

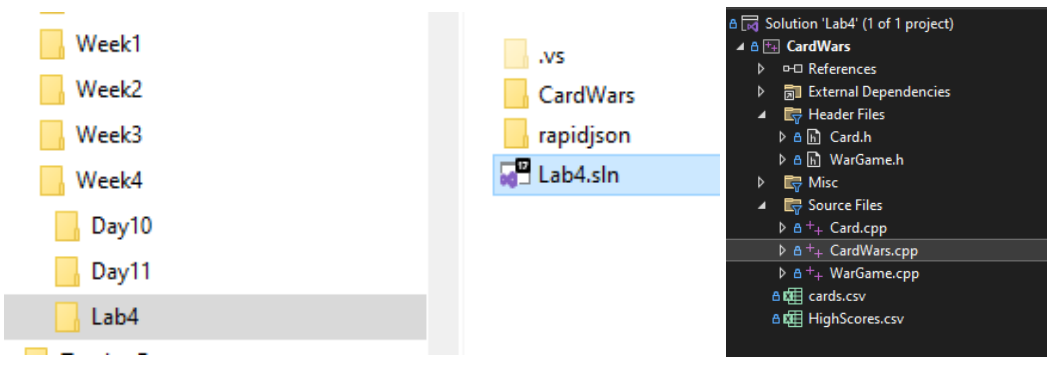
PG2 – LAB: CARD WARS

CONTENTS

COP2334

SETUP

A **C++ console application** has been provided for you in your **GitHub repo**. **Use the provided solution.**



Lab Video

Here's a video showing what the lab could look like when completed:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EdkuHdcMLoFCjT5ZMavtneIBNKjMgQrjhdPVitdlVx33Uw?e=Fphbky

How should you proceed?

The typical process for approaching the labs is…

Work on each part in order.

1. Watch the lecture video for Part A.
    a. The lectures have ***optional*** coding challenges. If you have time, then attempt them because they will help explain what you need to do for the lab.
2. Start working on Part A for the lab (Parts A-1 through A-3).
3. Once done with Part A, move on to Part B and repeat the steps above.

What to do if you need help?

Don't struggle for too long (more than 1 hour). Reach out for help to get your questions answered.

- Take advantage of the **open lab sessions** (they are in the evening so check the announcement channel in Discord for the zoom link). Here is the schedule: https://fullsailedu.sharepoint.com/:x:/s/emergingtechstu/EbBgblouNORLsms4V4hpAYgB0Mejv59w8IVGADkVgfJW8Q?e=0l2svR
- Ask your questions in the **PG2 Discord channels**. There will be instructors and students online who can help.
- Check the availability of **Full Sail tutors**. (https://discord.com/invite/8nV8PBqq7z)

Committing your code

You MUST commit and push your code to GitHub after completing each section. For instance, part A of this lab has 3 sections. You must commit and push after completing each section which would mean you end up with 3 separate commits for part A.

If you do not commit and push for every section, you could be deducted up to 15 points for the lab.

PART A: WORKING WITH CSV DATA

Lecture Videos for Part A

Day 10: File Input / Output Lecture:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EQqUwFqUvclApjsO4LPlErIBH5tplSrtuXhR_KzELpiS0A?e=l11TjC

Part A-1: Loading Cards

Add a static **LoadCards** method to the **WarGame** class. **Declare your method in the WarGame.h file and define the method in the WarGame.cpp file (**C++ Class Methods (w3schools.com)**).** It should have a string parameter for the file path. In the method, it should read the **cards.csv** file and **split** the data to get each string. Store each card in the _cards vector field of the class.

NAME	RETURNS	PARAMETERS	COMMENTS
LoadCards	nothing	string	Read the csv file and split the data.

Call the LoadCards method from the **WarGame constructor**.

Make sure to commit + push after completing the section.

Part A-2: Showing Cards

Add a static **ShowCards** method to the **WarGame** class. **Declare your method in the WarGame.h file and define the method in the WarGame.cpp file (**C++ Class Methods (w3schools.com)**).** Loop over the vector and call the Print method on each card.

NAME	RETURNS	PARAMETERS	COMMENTS
ShowCards	nothing	nothing	Prints each card that is in the vector.

In case 1 of the menu, call the ShowCards method **from main** (which is in the **CardWars.cpp** file).

Output the cards like the format as shown below.

```
1. Show Cards
2. Show High Scores
3. Play Card Wars
4. Exit

Choice? 1
A♥
2♥
3♥
4♥
5♥
6♥
7♥
8♥
9♥
10♥
J♥
Q♥
K♥
A♦
2♦
3♦
4♦
5♦
```

Make sure to commit + push after completing the section.

Part A-3: HighScore class

Add a **HighScore** class with a name **field** and a score **field** both with **getter/setter methods**.

Make sure to commit + push after completing the section.

Part A-4: Player class

Add a **Player** class with a **name** field, a **score field,** a pile field which is a vector of cards, and a won field which is a vector of cards. Add **getter/setter methods** for the score and name fields.

Add the following methods: **HasCards**, **PushCard**, **PopCard,** and **WonCards**. **HasCards** will return true if there are cards in the pile vector. **PushCard** will take a card parameter and add it to the pile vector of cards. **PopCard** will remove a card from the pile vector of cards and return the card. **WonCards** will take a vector of cards and add it to the won vector of cards and update the score field.

Make sure to commit + push after completing the section.

COP2334

PART B: WARGAME

Lecture Videos for Part B

Day 11: Serializing Lecture:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EcPrVLZJxURJhUXDM02szVcBZcdrIxaHz7ysJRNVhjCJfg?e=6n7YCA

Part B-1: Loading HighScores

Add 2 methods and a constructor to your HighScore class.

Add a **Deserialize** method that has 2 parameters: a string for the csv data and a char for the delimiter. The method should use the delimiter to get the name and score out of the csv parameter.

Add a HighScore **constructor** that calls the Deserialize method. It will need the same parameters as the Deserialize method.

Add a static **LoadHighScores** method to your **HighScore** class. It should have a string parameter for the file path. In the method, it should **open the HighScores.csv file** and read each line. It needs to create a HighScore object using the constructor you just created. Add the HighScore objects to a vector. Return the vector.

The file format: 1 high score object per line. Open the HighScores.csv file in the project to check the format.

NAME	RETURNS	PARAMETERS	COMMENTS
<constructor>		String Char	Calls the Deserialize method
Deserialize	Nothing	String char	The string parameter is the csvData that needs to be parsed. The char parameter is the expected delimiter in the csvData string.
LoadHighScores	vector<HighScore>	string	Deserializes the file into a vector and returns the vector.

In the main method in CardWars.cpp, before the do-while loop, call LoadHighScores passing the HighScoreFile variable and store the vector it returns into a variable to be used later.

Make sure to commit + push after completing the section.

Part B-2: Showing HighScores

Add a static **ShowHighScores** method to your **HighScore** class. It should have a vector<HighScore> const ref parameter. It should print a “High Scores” title then loop over the high scores vector and print each item. **Format the output so that the score has a color different than the name**. See example screenshot.

NAME	RETURNS	PARAMETERS	COMMENTS
ShowHighScores	nothing	vector<HighScore> const reference	Prints the vector of high scores.

In case 2 of the menu switch **in main** in **CardWars.cpp**, call the ShowHighScores method and pass the vector of highscores.

Output the high scores like the format as shown below.

```
1. Show Cards
2. Show High Scores
3. Play Card Wars
4. Exit

Choice? 2
----HIGH SCORES----
Batman            52
Bruce Wayne       52
Robin             40
Joker             39
GFG               32
Garrett           32
Alfred            31
Flash             30
Superman          29
Wonder Woman      28
Press ENTER to continue...
```

Make sure to commit + push after completing the section.

Part B-3: Saving HighScores

Add a **Serialize** method to the HighScore class. The method should have 2 parameters: an ofstream and a char. Using the ofstream, write the name and score to the file separating them with the char delimiter parameter.

Add a static **SaveHighScores** method to your **HighScore** class. It should have a string parameter for the file path and a vector<HighScore> const ref parameter. In the method, it should call the **Serialize** method of each HighScore in the vector. Call this method in the game when a player gets a new high score.

The file format: store 1 high score object per line. Open the HighScores.csv file in the project to check the format.

NAME	RETURNS	PARAMETERS	COMMENTS
Serialize	Nothing	Ofstream char	The ofstream is the file stream to write to. The char is the delimiter to use to separate the data.
SaveHighScores	nothing	string vector<HighScore> const reference	Serializes the vector into the file.

Make sure to commit + push after completing the section.

Part B-4: PlayGame

Add a **PlayGame** method to the **WarGame** class. **Declare your method in the WarGame.h file and define the method in the WarGame.cpp file (**C++ Class Methods (w3schools.com)**).** It should have 3 parameters: string for the player name, vector<HighScore> reference for the high scores, and a string for the name of the high score file.

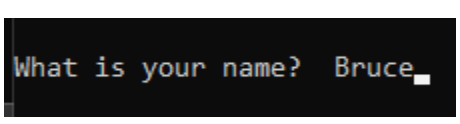
NAME	RETURNS	PARAMETERS	COMMENTS
PlayGame	nothing	Player's name String vector<HighScore> reference highscore file name string	See below for the description of what the method should do.

Here is the game logic to put in **PlayGame**:

1. Call the shuffle method to shuffle the vector of cards.
2. Create 2 instances of your Player class: npc and player. Give npc the name of "NPC". Use Input::GetString in main to get the name. Pass the name as a parameter to PlayGame and use it to set the name of the player instance.
3. Take the shuffled vector and add half of the deck to npc and half of the deck to the player.
4. Create another vector called **unclaimedPile**.
5. Loop while the player instance has cards
    a. Pop a card from the player instance and pop a card from the npc instance.
    b. Add the 2 cards to the unclaimed pile.
    c. Call the `Compare method of the card class`.
        i. NOTE: `Compare` will return -1 if the card1 < card2, 0 if card1 = card2, 1 if card1 > card2
        ii. Example of calling Compare: `int compResult = card1.Compare(card2);`
    d. If `Compare` returns -1, add the unclaimed pile to the npc instance. Clear the unclaimed pile. Print NPC wins.
    e. If `Compare` returns 1, add the unclaimed pile to the player instance. Clear the unclaimed pile. Print player wins. NOTE: use the player's name.
6. After the loop, check who won. Print the scores from the npc and player instances.
    a. If the npc has a higher score, print that the npc won the round.
    b. If the npc has the same score as the player, print that it was a tie.
    c. If the player has a higher score. Print out that the player won and check if it's a new high score.
        i. NOTE: the last score in the high score vector is the smallest high score. Therefore, if the player's score is greater than the last score in the high score vector, the player has a new high score.
        ii. If the player's score is a new high score,
            1. loop over high score vector to find the spot for the new high score.
                a. insert a new high score object into the vector at that index
                b. remove the last score in the vector
                c. call SaveHighScores (see part B-3)
                d. Call ShowHighScores to display the new top 10.
7. Ask the player if they want to play again. Use Input::GetInteger to get a number from the user: 1 for yes and 2 for no.
8. Continue playing while the user selects yes. Stop playing if they choose no.

In case 3 of the menu switch in main in **CardWars.cpp, call PlayGame to play a game of war!**

Output the rounds like the format as shown below.



```
 5♥ vs.  K♦ NPC wins!
 A♠ vs. 10♣ NPC wins!
 4♥ vs.  5♠ NPC wins!
10♥ vs.  6♣ Bruce wins!
 9♣ vs.  9♥
 9♦ vs. 10♦ NPC wins!
 Q♠ vs.  8♥ Bruce wins!
 5♣ vs.  8♣ NPC wins!
 3♠ vs.  7♣ NPC wins!
 6♦ vs.  3♣ Bruce wins!
 K♥ vs.  3♦ Bruce wins!
 5♦ vs.  Q♦ NPC wins!
 4♦ vs.  4♣
 A♥ vs.  8♦ NPC wins!
 A♣ vs.  7♠ NPC wins!
 6♠ vs.  J♦ NPC wins!
 2♠ vs.  6♥ NPC wins!
 3♥ vs.  2♥ Bruce wins!
 A♦ vs.  K♣ NPC wins!
 7♥ vs.  2♦ Bruce wins!
 J♠ vs.  K♠ NPC wins!
 9♠ vs. 10♠ NPC wins!
 7♦ vs.  8♠ NPC wins!
 Q♣ vs.  Q♥
 4♠ vs.  2♣ Bruce wins!
 J♥ vs.  J♣
 NPC wins.  34 to 16

Do you want to play again? (1: Yes, 2: No)
```

```
 7♠ vs.  A♣ Bruce wins!
 8♦ vs.  6♣ Bruce wins!
 9♣ vs.  9♥
 K♦ vs.  Q♦ Bruce wins!
 7♦ vs.  3♥ Bruce wins!
 2♦ vs.  6♠ NPC wins!
 J♣ vs.  9♠ Bruce wins!
 4♠ vs.  5♣ NPC wins!
 3♦ vs.  A♠ Bruce wins!
 7♣ vs.  7♥
10♣ vs.  5♠ Bruce wins!
 6♦ vs.  4♥ Bruce wins!
 9♦ vs. 10♠ NPC wins!
 4♣ vs.  K♣ NPC wins!
 8♥ vs.  6♥ Bruce wins!
 8♠ vs.  3♣ Bruce wins!
 J♥ vs.  Q♠ NPC wins!
 3♠ vs. 10♦ NPC wins!
 4♦ vs.  2♥ Bruce wins!
 2♣ vs.  5♥ NPC wins!
 J♠ vs.  J♦
 K♥ vs.  A♥ Bruce wins!
 Q♣ vs.  A♦ Bruce wins!
 2♠ vs.  Q♥ NPC wins!
10♥ vs.  8♣ Bruce wins!
 K♠ vs.  5♦ Bruce wins!
 Bruce WINS!  36 to 16
_#_#_  NEW HIGH SCORE  _#_#_
----HIGH SCORES----
Batman          52
Bruce Wayne     52
Robin           40
Joker           39
Bruce           36
Garrett         32
Alfred          31
Flash           30
Superman        29
Wonder Woman    28

Do you want to play again? (1: Yes, 2: No)
```

Make sure to commit + push after completing the section.