PROGRAM 3 ASSIGNMENT INSTRUCTIONS

STACKS & QUEUES, CSC 1310-001, SPRING 2025

DUE DATE

Thursday, March 27, 2024

You may submit up to 2 days late with 10 points off per day late.

IMPORTANT RULES YOU MUST FOLLOW SO YOU DO NOT GET A ZERO

As reported in the syllabus, you may not use the following in your code:

- while(true) or while(1)
- Goto statements
- Recursively calling the main function
- exit()
- ternary operators (any syntax requiring a '?')

GENERATIVE AI POLICY FOR PROGRAM 3 ASSIGNMENT

- You may use generative AI **only to help you with troubleshooting the code**. **However, if you do, you must share your entire generative AI conversation**. Either take screenshots of your entire chat conversations, paste to a word document, and include with your submission, or if you have ChatGPT 4.0, you can share a public link to your chat by clicking "**Share**" in the top right, creating the link, and then creating a document where you paste all your chat links and make sure to upload that with your submission.
- **You may NOT copy and paste any part of the programming assignment instructions or sample output**, but you may upload the code to aid in troubleshooting.
- **You may NOT ask generative AI to create functions or any code for you.** This will be considered academic misconduct for this assignment.
- **You may not show more then 1 to 2 lines of code to another person** (parent, other students – even if they are not in our class) who is not a TA or instructor. This will be considered academic misconduct for this assignment.
- If you get ideas on how to write a portion of your code, you must **CITE YOUR SOURCE** in a comment in that section of your code. You still must be able to explain what the code is doing.

DESCRIPTION

You are implementing the game of War using a C++ program. You will use a dynamic queue to represent each player's deck and use a stack to store temporary piles of cards. Plyers will be able to draw cards from their respective deck, and the higher card wins.

GIVEN

You will be given a source file named given.cpp that contains most of the main function as well as all the functions that display ASCII art to show the different cards as well as the word "WAR", "PLAYER ONE WINS", and "PLAYER TWO WINS". You will rename this file to **War.cpp**.

SUBMISSION

You should submit the following files in a zipped folder to the program 3 assignment in ilearn.

- **Queue.h** (dynamic linked list queue class) – do not use the STL, you must create your own queue class
- **Stack.h** (dynamic linked list stack class) – do not use the STL, you must create your own stack class
- **Card.h** (Card class)
- **Game.h** (Game class with game logic)
- **War.cpp** (runs the game)

CARD.H

The Card class will allow you to create Card objects in your program. Your Queue and Stack class will have a Card object in each node that is created, so those classes will need to #include the Card header file.

The Card class contains two private **attributes**:

- an integer representing the card number (2 to 14 where 11 is Jack, 12 is Queen, 13 is King, and 14 is Ace)
- a character holding the suit of the card ('H' for hearts, 'D' for diamonds, 'C' for clubs, and 'S' for Spades)

You should have the following **member functions** in the Card class:

- **Card constructor** – sets cardNum to zero and suit to a space ' ' character
- **Card overloaded constructor** – sets cardNum and suit to the values sent to the function
- **2 mutator functions and 2 accessor functions**
- An **overloaded operator** function for the greater than "**>**" operator which will compare card values and return true if greater than and false otherwise.
- An **overloaded operator** function for the equal-to "**==**" operator, which will return true if they are equal and false otherwise.

QUEUE.H

The Queue class will be a dynamic queue implementation. The linked list Node should be created as a struct private attribute in this class and the node contains a Card object as the value.

You should have the following functions in the Queue class:

- **Queue constructor** – sets front & rear to NULL and size to zero
- **Queue destructor** – should delete all nodes left in the queue
- **Enqueue** (should accept a Card object as a parameter and create a new node that is appended to the back of the queue)
- **Dequeue** (should delete the front node & return the Card object that was in that node)
- **isEmpty** – should return true if there are no nodes in the queue and false otherwise
- **getSize** – should return the number of nodes in the queue

STACK.H

The Stack class will be a dynamic stack implementation. The linked list Node should be created as a struct private attribute in this class and the node contains a Card object as the value.

You should have the following functions in the Stack class:

- **Stack constructor** – sets top to NULL
- **Stack destructor** – should delete all nodes left in the stack
- **Push** (should accept a Card object as a parameter and create a new node that is placed on top of the stack)
- **Pop** (should delete the top node & return the Card object that was in that node)
- **isEmpty** – should return true if there are no nodes in the stack and false otherwise
- **getSize** – should return the number of nodes in the stack

WAR.CPP

Take the given.cpp source file and rename it to war.cpp. This file contains a mostly-written main function and all the helper functions that print ASCII art to the screen. You may modify these functions if you want, however, there are some specific things that you must do in order to get full credit on this assignment.

Each player's deck of cards should be stored in a Queue (so you will have two queues, which are already defined for you in the main function of the given source file).

You must create a **play()** function and a **resolveWar()** function. The description of what should happen in these functions is below and there are some details in the given file in comments.

PLAY FUNCTION

This function simulates the game (has the loop that keeps going until someone wins). A player wins win the other player runs out of cards (their queue becomes empty!)

The **play()** function must do the following:

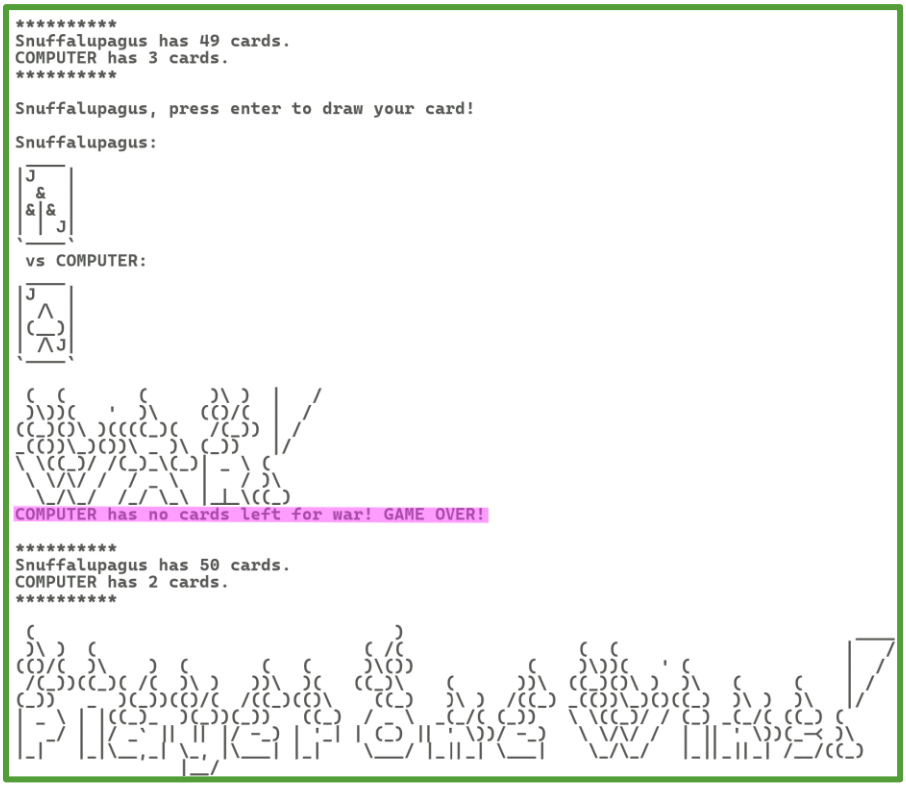
- Read in player 1's name. Player 2 is the computer.
- Remove (and save) a card from both player's queue (deck). Have the user press enter to simulate them drawing a card. Print the user's card to the screen by calling the **displayCard()** function. Then print the computer's card calling the **displayCard()** function again. Below is sample output if the user's name entered was "Snuffalupagus"

```
Player 1, what is your name?  Snuffalupagus
Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|J   |
|  /\|
|  \/|
|   J|
`____`
 vs COMPUTER:
 ____
|4   |
|(\/)|
|  V|
|   4|
`____`
```

- Check if player 1's card is greater than player 2's card. If so, then add both cards to player 1's queue. Otherwise, add both cards to player 2's queue. Whoever got the cards added to their queue, print who won the round like the sample output below.
- Also, print how many cards each player has.

```
Snuffalupagus won this round and gets both cards!

**********
Snuffalupagus has 27 cards.
COMPUTER has 25 cards.
**********
```

- If the two cards that are drawn are the same, then the players go to War! This is when you call the **resolveWar()** function.
    - The resolveWar function returns a Boolean that indicates if the game should be over or not. It would return "true" if one of the players ends up not having enough cards to resolve the war.
    - If it does return true, the game should end. (Example Output of this below)



RESOLVEWAR FUNCTION

This function handles a tie "war" situation using a Stack. Here is an example of when this would happen. Notice the sample output below shows both players getting a Queen card.

```
**********
Snuffalupagus has 24 cards.
COMPUTER has 28 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 _____
|Q    |
|(\/) |
| \/  |
|    Q|
`_____`
 vs COMPUTER:
 _____
|Q    |
|  /\ |
| (__)|
|  /\Q|
`_____`
```

Follow the steps below to write this function:

1. Call the **printWar()** function that was provided for you:

```
  (  (       (      )\ )  |    /
  )\))(   '  )\    (()/(  |   /
 ((_)()\ )(((_)(    /(_)) |  /
 _(())\_)())\ _ )\  (_))   | /
 \ \((_)/ /(_)_\(_)| _ \  (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!
```

2. Take the two cards sent to this function from the players (the ones that tied) and push them both to a Stack. I will call this Stack “warPile”.

3. **Check to see if either player’s deck is smaller than 4 cards**. If so, print that a player has no cards left for war and print that the game is over. When the resolveWar function ends, it should return “true” to indicate the game is over.

4. Remove 3 cards from each player’s deck (Queue) and push them into the warPile (Stack).

5. Then, take one more card from each player’s deck (Queue) and save each to their own temporary Card variable so that you can compare them. Also, push these two cards to the warPile (Stack). Call the displayCard() function on these cards. Example output is below:

```
Snuffalupagus:
 _____
|5    |
|  /\ |
| (__)|
|  /\5|
`_____`
 vs COMPUTER
 _____
|4    |
|  /\ |
| (__)|
|  /\4|
`_____`
```

6. Resolve:
    a. If player 1's war card is greater than player 2's, then add all the cards from the war pile to player 1's deck (Queue).

```
Snuffalupagus, press enter to draw your WAR card!

Snuffalupagus:
 ____
|K   |
|(\/) |
| \/ |
|   K|
`____`
 vs COMPUTER
 ____
|6   |
|(\/) |
| \/ |
|   6|
`____`


Snuffalupagus won this WAR round and gets all cards!
```

    b. Otherwise, if they are the same value, **recursively call resolveWar()** sending the war cards drawn. Before calling resolveWar recursively, print out DOUBLE WAR! Like you see below:

```
*************************************************************************************************
****************************     DUPLICATE WAR!!!!!!     ********************************************
*************************************************************************************************
 (  (       (      )\ )  |    /
 )\))(   '   )\    (()/(  |   /
((_)()\ )((((_)(   /(_)) | /
_(())\_)())\ _ )\ (_))   |/
\ \((_)/ /(_)_\(_)| _ \ (
 \ \/\/ /  / _ \  |   / )\
  \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!
```

    c. Otherwise, that means player 2's war card is greater than player 1's, so then add all the cards from the war pile to player 2's deck (Queue)

```
Snuffalupagus:
 ____
|7   |
| /\ |
| \/ |
|   7|
`____`
 vs COMPUTER
 ____
|A   |
| /\ |
|(__)|
| /\A|
`____`


COMPUTER won this WAR round and gets all cards!
```

TESTING – ARE THERE 52 CARDS?

I highly recommend you test multiple times.

Each time, **make sure that when you add the number of cards in both player's decks, that it is always 52**. If not, then there is probably an issue with how you wrote the resolveWar() function.

Also, make sure there are only 52 cards left after calling War and one of the players runs out of cards.

SAMPLE OUTPUT

Originally, I tried to paste the entire output of playing one game below and this document became 125 pages, which is ridiculous. Instead, I removed most of the pages of output in the middle so you could get the gist of what the sample output should be.

```
Shuffling and dealing the deck! There are 52 cards.

Player 1, what is your name?  Snuffalupagus
Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|10  |
| /\ |
| \/ |
|  10|
`----`
 vs COMPUTER:
 ____
|2   |
| /\ |
| \/ |
|   2|
`----`


Snuffalupagus won this round and gets both cards!

**********
Snuffalupagus has 27 cards.
COMPUTER has 25 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|3   |
|(\/)|
| \/ |
|   3|
`----`
 vs COMPUTER:
 ____
|K   |
|(\/)|
| \/ |
|   K|
`----`


COMPUTER won this round and gets both cards!

**********
Snuffalupagus has 26 cards.
COMPUTER has 26 cards.
**********

Snuffalupagus, press enter to draw your card!
```

```
Snuffalupagus:

 _____
|3    |
| /\  |
|(__)|
| /\3|
`----`
 vs COMPUTER:

 _____
|6    |
| /\  |
|(__)|
| /\6|
`----`


COMPUTER won this round and gets both cards!

**********
Snuffalupagus has 25 cards.
COMPUTER has 27 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:

 _____
|Q    |
|(\/)|
| \/ |
|    Q|
`----`
 vs COMPUTER:

 _____
|J    |
| /\  |
| \/ |
|    J|
`----`


Snuffalupagus won this round and gets both cards!

**********
Snuffalupagus has 26 cards.
COMPUTER has 26 cards.
**********
```

----------BUNCHES OF LINES OF OUTPUT WAS REMOVED HERE----------

```
**********
Snuffalupagus has 18 cards.
COMPUTER has 34 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:

 _____
|6    |
```

```
|(\/)|
| \/ |
|   6|
`----`
 vs COMPUTER:

 ____
|6   |
| /\ |
| \/ |
|   6|
`----`

  (  (       (      )\ )  |   /
  )\))(   '  )\    (()/(  |  /
 ((_)()\ )((((_)(   /(_)) | /
 _(())\_)())\ _ )\ (_))   |/
 \ \((_)/ /(_)_\(_)| _ \ (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!

Snuffalupagus:

 ____
|10  |
| &  |
|&|& |
| |10|
`----`
 vs COMPUTER

 ____
|10  |
| /\ |
|(__)|
| /10|
`----`


**********************************************************************************************
**************************     DUPLICATE WAR!!!!!!     ***********************************
**********************************************************************************************
  (  (       (      )\ )  |   /
  )\))(   '  )\    (()/(  |  /
 ((_)()\ )((((_)(   /(_)) | /
 _(())\_)())\ _ )\ (_))   |/
 \ \((_)/ /(_)_\(_)| _ \ (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!

Snuffalupagus:

 ____
|10  |
| /\ |
| \/ |
|  10|
`----`
 vs COMPUTER

 ____
|7   |
| /\ |
|(__)|
| /\7|
```

```
`----`


Snuffalupagus won this WAR round and gets all cards!

**********
Snuffalupagus has 27 cards.
COMPUTER has 25 cards.
**********
```

----------BUNCHES OF LINES OF OUTPUT WAS REMOVED HERE----------

```
**********
Snuffalupagus has 6 cards.
COMPUTER has 46 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:

 ____
|8   |
|(\/)|
| \/ |
|   8|
`----`
 vs COMPUTER:
 ____
|9   |
| &  |
|&|& |
| | 9|
`----`


COMPUTER won this round and gets both cards!

**********
Snuffalupagus has 5 cards.
COMPUTER has 47 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:

 ____
|7   |
| /\ |
| \/ |
|   7|
`----`
 vs COMPUTER:
 ____
|J   |
| &  |
|&|& |
| | J|
`----`
```

```
COMPUTER won this round and gets both cards!

**********
Snuffalupagus has 4 cards.
COMPUTER has 48 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|5   |
| /\ |
| \/ |
|   5|
`----`
 vs COMPUTER:
 ____
|3   |
| /\ |
| \/ |
|   3|
`----`


Snuffalupagus won this round and gets both cards!

**********
Snuffalupagus has 5 cards.
COMPUTER has 47 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|3   |
| /\ |
|(__)|
| /\3|
`----`
 vs COMPUTER:
 ____
|K   |
|(\/)|
| \/ |
|   K|
`----`


COMPUTER won this round and gets both cards!

**********
Snuffalupagus has 4 cards.
COMPUTER has 48 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
```

```
 ____
|8   |
| &  |
|&|& |
| | 8|
`----`
 vs COMPUTER:

 ____
|2   |
| &  |
|&|& |
| | 2|
`----`


Snuffalupagus won this round and gets both cards!

**********
Snuffalupagus has 5 cards.
COMPUTER has 47 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:

 ____
|5   |
|(\/)|
| \/ |
|   5|
`----`
 vs COMPUTER:

 ____
|5   |
| /\ |
|(__)|
| /\5|
`----`

  (  (       (      )\ )  |   /
  )\))(   '  )\    (()/(  |  /
 ((_)()\ )((((_)(   /(_)) | /
 _(())\_)())\ _ )\ (_))   |/
 \ \((_)/ /(_)_\(_)| _ \ (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!

Snuffalupagus:

 ____
|2   |
| &  |
|&|& |
| | 2|
`----`
 vs COMPUTER

 ____
|K   |
| &  |
|&|& |
| | K|
`----`
```

```
COMPUTER won this WAR round and gets all cards!

**********
Snuffalupagus has 0 cards.
COMPUTER has 52 cards.
**********

 (                                                                          ____
 )\ )  (                          *   )               (  (                    |   /
(()/(  )\    )  (       (   (    ` )  /( (  (          )\))(   ' (            |  /
 /(_))((_)( /(  )\ )   ))\  )(    ( )(_)))\))(    (   ((_)()\ )  )\   (     (    | /
(_))    _  )(_))(()/(  /((_)(()\  (_(_())((_)()\   )\  _(())\_)()((_)  )\ )  )\   |/
| _ \ | |((_)_  )(_))(_))   ((_) |_   _|_(()((_) ((_) \ \((_)/ / (_) _(_/( ((_) (
|  _/ | |/ _` || || |/ -_) | '_|   | |  \ V  V // _ \  \ \/\/ /  | || ' \))(_-< )\
|_|   |_|\__,_| \_, |\___| |_|     |_|   \_/\_/ \___/   \_/\_/   |_||_||_| /__/((_)
                |__/
```

SAMPLE OUTPUT 2

I am only placing the ending to another running of this game because I wanted to demonstrate what the resolveWar() function should do if a player doesn't have enough cards left for War. This happened to also happen in "Duplicate War". This is something you should test so you may have to run multiple times before this happens.

```
**********
Snuffalupagus has 11 cards.
COMPUTER has 41 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|8   |
| &  |
|&|& |
| | 8|
`----`
 vs COMPUTER:
 ____
|8   |
|(\/)|
| \/ |
|   8|
`----`

  (  (       (      )\ )  |   /
  )\))(   '  )\    (()/(  |  /
 ((_)()\ )((((_)(   /(_)) | /
 _(())\_)())\ _ )\ (_))   |/
 \ \((_)/ /(_)_\(_)| _ \ (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!

Snuffalupagus:
```

```
 ____
|5   |
| /\ |
| \/ |
|   5|
`----`
 vs COMPUTER
 ____
|9   |
| &  |
|&|& |
| | 9|
`----`


COMPUTER won this WAR round and gets all cards!

**********
Snuffalupagus has 6 cards.
COMPUTER has 46 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|Q   |
| &  |
|&|& |
| | Q|
`----`
 vs COMPUTER:
 ____
|8   |
| /\ |
| \/ |
|   8|
`----`


Snuffalupagus won this round and gets both cards!

**********
Snuffalupagus has 7 cards.
COMPUTER has 45 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|4   |
| /\ |
|(__)|
| /\4|
`----`
 vs COMPUTER:
 ____
|A   |
| /\ |
|(__)|
| /\A|
```

```
`----`


COMPUTER won this round and gets both cards!

**********
Snuffalupagus has 6 cards.
COMPUTER has 46 cards.
**********

Snuffalupagus, press enter to draw your card!

Snuffalupagus:
 ____
|5   |
| /\ |
|(__)|
| /\5|
`----`
 vs COMPUTER:
 ____
|5   |
|(\/)|
| \/ |
|   5|
`----`

  (  (       (      )\ )  |   /
  )\))(   '  )\    (()/(  |  /
 ((_)()\ )((((_)(   /(_)) | /
 _(())\_)())\ _ )\ (_))   |/
 \ \((_)/ /(_)_\(_)| _ \ (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
Snuffalupagus, press enter to draw your WAR card!

Snuffalupagus:
 ____
|Q   |
| &  |
|&|& |
| | Q|
`----`
 vs COMPUTER
 ____
|Q   |
| /\ |
| \/ |
|   Q|
`----`


**********************************************************************************************
**************************     DUPLICATE WAR!!!!!!     **********************************
**********************************************************************************************
  (  (       (      )\ )  |   /
  )\))(   '  )\    (()/(  |  /
 ((_)()\ )((((_)(   /(_)) | /
 _(())\_)())\ _ )\ (_))   |/
 \ \((_)/ /(_)_\(_)| _ \ (
  \ \/\/ /  / _ \  |   / )\
   \_/\_/  /_/ \_\ |_|_\((_)
```

```
Snuffalupagus has no cards left for war! GAME OVER!
Snuffalupagus has no cards left for DUPLICATE WAR! GAME OVER!

**********
Snuffalupagus has 1 cards.
COMPUTER has 51 cards.
**********
```