Assignment 2

1 Assignment Policy

- Weight: this assignment is worth 20% of your final grade.
- Grading: out of 33 points, as explained below.
- Due Date: Dec 6, 2015 11:50PM
- Late Submission policy: 10% penalty up to five days after the due date. NO ASSIGNMENT WILL BE ACCEPTED 5 days after the deadline.
- Deliverables: One Microsoft Word document and one Visual Studio project (as explanined below) zipped in a single zip file using this naming convention: if your name is John Smith, the file name must be JohnSmithA2.zip. The required file must be posted on Blackboard in the Assignment 2 folder created for this purpose.
- The submitted work must be individual. All submissions will be compared using software and if two submissions are similar, they will be punished according to George Brown Academic Honesty Policy.

2 The card game of War

One of the easiest card games to learn is the game of War. A standard deck of playing cards is shuffled and dealt to two players. Each player stars with a hand of 26 cards. The players repeatedly battle with their cards. A battle consists with of each player placing the *top card* from his or her hand face up on the table. Whoever has the higher of the two cards wins the battle prize (the two cards) and places those cards at the *bottom of their hands*. **Question: What does that mean in terms of Data Structures?** A tied battle means war!
In a war each player adds three more cards to the prize pile and then turns up another battle card. Whoever wins this battle gets to add the entire prize pile - all 10 cards - to the bottom of his/her hand. If there is another tie, another war follows and the prize pile grows larger. The game continues until one of the players runs out of cards. That player loses.

3 Simulation

For the purpose of this assignment, you have to simulate a number of games, and compute the average number of battles waged in each game. This gives us an idea of how long a game takes, whichmay influence our decision to play.
To simulate War, we just have to deal cards randomly to the two hands. **Both hands must be represented by two queues.** Them youmust coordinate the enqueuing and dequeuing operations among the two hands and the prize pile according to the rules of the game.
Our program requires two input values: the number of games to simulate and themaximun number of battles allowed before a game is discontinued. The latter value is needed to ensure that your program does not run forever. There is no guarantee that a game of War will ever end! Output from the program consists of statistics about the number of discontinued games, the number of completed games, and the average number of battles waged in completed games.

4 The deliverables

- (10 POINTS) The `RankDeck class` should model a deck of cards. It should contain as minimum a container for the cards (chose the best structure!), a constructor, a shuffle method, a dealNextrCard method, and a boolean method that indicates if there are more cards to be dealt.

- (15 POINTS) The `WarGame class` simulates a game of War. Its constructor takes an argument indicating the maximum number of battles permitted before discontinuing a game.The class must have a `play` method that simulates a game until is finished or dicontinued, retuning `true` if the game finishes normally of `false` if it is discontinued. It also must contain a method that returns the number of battles waged inthe most recent game.
  The `play` method, when invoked, must create **three queues**, two for each player's hand, and one for the prize pile of cards. Next, the method shuffles the deck and deals the crads to the players, that is it enqueues (push!) them. Once the cards are dealt, the `play` method calls the recursive `battle` method, which enacts the battle between the players. This cycle continues until the game is over or dicontinued. The algorithm for the `battle` is below:

  ```
  battle()
  ```

```
Get player1's card from player1's hand
Put player1's card in the prize pile
Get player2's card from player1's hand
Put player2's card in the prize pile
if (player1's card > player2's card)
    remove all the cards from the prize pile and put them in player2's hand
else //war!
    each player puts three cards in the prize pile
    battle()
```

- (5 POINTS) The `main()` program implements user input and presents the results. Example:

```
How many games shouldbe simulated? 1000
What is the maximum number of battles per game? 300

Number of games simulated: 1000
Number of discontinued games: 658
Number of completed games: 342

In the completed games:
   Total number of battles: 56138
   Avergae number of battles: 164

Program completed.
```

- The textbook code for the queues has been provided.