University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 161 Name: `war.cpp`
Project 5 Due: see http://ece161.viall.org

Write a program which emulates the card game of "War". The program should prompt the user for a "seed" which is used to initialize the random number generator. This will determine the initial shuffled order of the deck of cards.

A sample run might look as follows:

```
WAR card game v1.0
Enter seed:2
A:26,B:26 A:8♠-B:7♠ A Wins
A:27,B:25 A:J♠-B:T♥ A Wins
A:28,B:24 A:A♥-B:6♥ A Wins
A:29,B:23 A:Q♠-B:5♦ A Wins
A:30,B:22 A:2♦-B:4♠ B Wins
A:29,B:23 A:6♠-B:J♦ B Wins
A:28,B:24 A:T♠-B:T♦ !!WAR!! [A:7♥,B:8♣; A:K♣,B:3♠; A:4♣,B:9♦]
A:24,B:20 A:7♣-B:4♦ A Wins
A:33,B:19 A:6♦-B:9♠ B Wins
            :
A:39,B:13 A:3♦-B:9♠ B Wins
A:38,B:14 A:Q♣-B:5♦ A Wins
A:39,B:13 A:9♥-B:9♣ !!WAR!! [A:K♥,B:8♣; A:2♠,B:J♣; A:T♣,B:3♠]
A:35,B: 9 A:8♥-B:J♥ B Wins
A:34,B:18 A:Q♥-B:4♣ A Wins
              :
A:29,B:23 A:6♦-B:J♣ B Wins
A:28,B:24 A:Q♠-B:T♣ A Wins
A:29,B:23 A:3♣-B:3♠ !!WAR!! [A:T♠,B:8♥; A:2♥,B:J♥; A:T♦,B:3♥]
A:25,B:19 A:6♣-B:K♠ B Wins
A:24,B:28 A:7♥-B:7♦ !!WAR!! [A:4♥,B:9♦; A:K♣,B:5♣; A:2♣,B:A♦]
A:20,B:24 A:7♣-B:5♠ A Wins
A:29,B:23 A:5♥-B:8♦ B Wins
A:28,B:24 A:A♠-B:2♦ A Wins
A:29,B:23 A:T♥-B:9♠ A Wins
A:30,B:22 A:K♦-B:7♠ A Wins
A:31,B:21 A:J♦-B:9♥ A Wins
A:32,B:20 A:Q♦-B:4♠ A Wins
A:33,B:19 A:6♥-B:9♣ B Wins
A:32,B:20 A:Q♣-B:J♠ A Wins
            :
A:40,B:12 A:8♠-B:T♦ B Wins
A:39,B:13 A:5♦-B:K♥ B Wins
A:38,B:14 A:K♣-B:3♣ A Wins
A:39,B:13 A:4♣-B:8♣ B Wins
A:38,B:14 A:A♦-B:7♦ A Wins
A:39,B:13 A:4♦-B:J♣ B Wins
A:38,B:14 A:9♣-B:9♦ !!WAR!! [A:3♦,B:A♥; A:7♣,B:3♥; A:7♠,B:K♠]
A:34,B:10 A:5♠-B:5♥ !!WAR!! [A:9♥,B:8♦; A:A♠,B:8♠; A:3♠,B:T♦]
A:30,B: 6 A:2♦-B:5♦ B Wins
A:29,B:23 A:T♠-B:K♥ B Wins
A:28,B:24 A:T♥-B:4♣ A Wins
A:29,B:23 A:2♠-B:8♣ B Wins
A:28,B:24 A:9♠-B:4♦ A Wins
A:29,B:23 A:8♥-B:J♣ B Wins
              :
A:41,B:11 A:J♥-B:A♥ B Wins
A:40,B:12 A:4♥-B:4♣ !!WAR!! [A:Q♥,B:K♠; A:Q♦,B:4♦; A:3♥,B:8♦]
A:36,B: 8 A:5♣-B:3♦ A Wins
A:45,B: 7 A:A♠-B:T♦ A Wins
A:46,B: 6 A:Q♣-B:5♥ A Wins
A:47,B: 5 A:7♠-B:K♥ B Wins
A:46,B: 6 A:Q♠-B:7♣ A Wins
A:47,B: 5 A:5♠-B:8♣ B Wins
A:46,B: 6 A:J♠-B:J♥ !!WAR!! [A:3♠,B:A♥; A:9♥,B:7♠; A:2♦,B:K♥]
A:42,B: 2 A:T♣-B:5♠ A Wins
A:51,B: 1 A:T♠-B:8♣ A Wins
[** B out of cards **]
We have a winner
Congratulations Player A
Press any key to continue
```

RULES FOR THE CARD GAME WAR:

General

1. The standard 52 card deck is used. Card rank from high to low is A K Q J T 9 8 7 6 5 4 3 2. Suits are ignored for this game (but your program must still keep track of them.

2. The deck is shuffled and all cards are dealt to the two players (each player will have 26 cards).

3. The object of the game is to win all the cards.

The Play

4. Each player places the top card from their "pile" face up on the table.

5. Whichever player put down the highest ranking card takes both cards and places them on the bottom of their pile.

6. Steps 4 and 5 are repeated until one player runs out of cards (in which case the other player wins), or both played cards are of equal rank.

War

7. If both cards from step 4 (or step 9) are of equal rank, there is a WAR. Each player places three additional cards from their "pile" face down on the table.

8. Each player then places the top card from their "pile" face up on the table.

9. Whichever player put down the highest ranking card takes all of the cards on the table (10 cards in the case of a single war, 18 cards for a double war, 26 cards for a triple war, etc) and places them at the bottom of their pile. If both of the cards played in step 8 are of equal rank, go to rule 7.

Winning

10. If at any point during the game a player is required to play a card and is unable to (because the player has no cards), then that player looses the game, and the other player wins. Note that it is possible for a player to play the last card in their "pile", win that hand, and then continue.

[Thanks to http://www.pagat.com/war/war.html which was a help in articulating these rules]

Programming Considerations:

Let each Players "pile" be a queue. Cards on the "table" are also a queue.

The Play:

```
cardA = dequeue(pileA)
enqueue(table) = cardA
cardB =dequeue(pileB)
enqueue(table) = cardB
if cardA > cardB then
        // A wins
        card = dequeue(table)              while notEmpty(table)
        enqueue(pileA) = card      or            card=dequeue(table)
        card = dequeue(table)                    enqueue(pileA)=card
        enqueue(pileA) = card
else if cardA < cardB then
        // B wins
        card = dequeue(table)              while notEmpty(table)
        enqueue(pileB) = card      or            card=dequeue(table)
        card = dequeue(table)                    enqueue(pileB)=card
        enqueue(pileB) = card
else
        // it's a war
        for i=1 to 3
            card=dequeue(pileA)
            enqueue(table)=card
            card=dequeue(pileB)
            enqueue(table)=card
```

Coding:

Consider using a simple array with "wrap around" front and rear pointers. Note the queue_size must be 53 (not 52).

An empty_queue condition occurs when: front==rear;
A full_queue condition occurs when: front==(rear+1)%53

A win occurs when the other player has a queue_empty condition. You must also check to see if the other player has a queue_empty condition to cover the rare instance of a "tie"; that is when both players have an equal number of cards and have a sextuple war (i.e. 6 repeated wars).

Each card could be stored in one integer value – the lower byte could be the rank (or value) of the card; the next byte could be the suit. Two arrays could be used to help with output:

```
char cRank[]={'?','?','2','3','4','5','6','7','8','9','T','J','Q','K','A'};
char cSuit[]={ '\x03', '\x04', '\x05', '\x06' };
```

To print a card as a rank and suit, you could say:

```
printf("%c%c", cRank[somecard&0xff], cSuit[somecard>>8]);
```

Alternatively, you may use a structure with fields `.rank` and `.suit`. For example:

```
typedef struct card { char rank; char suit; } CARD;
CARD deck[52];  // 52 ok for deck; Q's must be 53
```

Coding (continued)

Creating and shuffling the deck is accomplished by employing a random number seed. The pertinent pieces of code are:

```
#include <stdlib.h>      // .h file for rand()

int seed,i,j,suit,rank;
int randompile[52], sortedpile[52];
// note these may be 52, but queues must be 53

printf("Enter seed: ");
scanf("%d",&seed);
srand(seed);   // "seed" random generator

i=0;
for (suit=0; suit<=3; suit++)
     for (rank=2; rank<=14; rank++)
          sortedpile[i++]=suit<<8|rank;
for (i=0; i<52; i++)
{
     j=rand()%(52-i);
     randompile[i]=sortedpile[j];
     sortedpile[j]=sortedpile[52-1-i];
}
```

Extra Credit

Part one and part two are each worth 100 points. You may do either or both. Maximum possible score for war.cpp is 300 points

Part 1:

Implement this game using recursive function call whenever a war happens; if repeated wars continue to happen, the function should go to a new level for each war. For example, a triple war should result in 3 levels of calls to the recursive function.

Part 2:

Detect and report as soon as possible when the game will go on indefinitely. This has to be more than a simple "you've done this 100,000 times, it must go on forever" type of detection. One possible way is to keep track of the order of cards in each players hand...if the order repeats exactly for some number of plays, then the game will never end...otherwise known as Lamb Chop mode.