Project 1
Old Maid (Card Game)

Name: Tri Le
Instructor: Dr. Lehr
Section: Spring 2016 CSC17C - 42102
Date: 11 May 2016

Introduction

Game Program: Old Maid

What is it?: Old Maid is a card game that that uses a standard deck of 52 playing cards. The deck is shuffled before beginning the game and one queen, it does not matter what suit it is, is removed from the deck so that there are only 3 queens total in the deck of cards. The cards are then evenly distributed to all the players. Although, one player will have 1 less card than everyone else, but this is fine. After all the players receive their cards, they throw away all pairs they have. After this is done, the players go in either a clockwise or counter clockwise direction and they face the player depending on which order they chose and they let the other player blindly take a card from their hand. Any pairs obtained during this time are to be thrown away and this is repeated until only one player is left with a queen. That person is deemed to be the loser. The program was made by first using a deque to create 51 cards in ascending order, 51 cards because we have to take into account the queen we are removing before starting. The cards are labeled by their numbers and Jacks are represented as 11's, Queens are represented by 12's and Kings are 13's. There are 4 of each card in the deck except the queen which only has 3. The suits do not matter in Old Maid so I did not use them to identify any given cards, so all of the cards in the deque are their numbers four times in the deck. i.e. {1,1,1,1,2,2,2,2, etc.}. The deque is then shuffled and after that it is pushed into a stack. From the stack the cards are popped off onto the player's hands, which are lists. After that, all pairs are thrown away. The game begins here, player's go in a counter clockwise direction and take cards from the other player that is "technically" to their left. Player 1 will take from player 3, player 2 will take from player 3 and player 3 will take from player 1. All pairs are to be thrown away during the process of taking and this is repeated until one player is left with the queen(12). The person with the queen is the loser of the game.

Summary

Statistics of the program:
Total number of lines: 380 lines
Lines in main.cpp: 242 lines
Lines in Hand.h: 35 lines
Lines in Hand.cpp: 103 lines
Number of total functions: 13 functions
Number of functions in main.cpp: 4 functions
Number of functions in Hand class: 9 functions
Number of total objects used: 6 objects
Objects used: std::map, std::deque, std::stack, std::list, self created object
Number of total variables: 13 variables

This project utilized the STL's contents that were required for this project. The project met the minimum line count of 300 with it being at 380 lines.

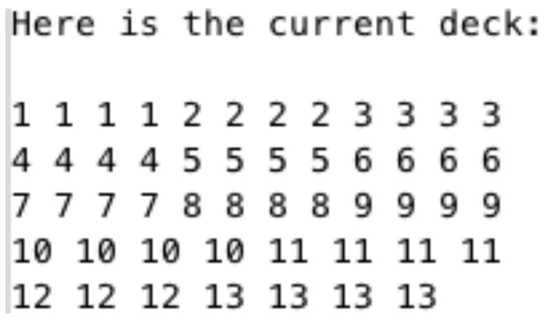
This project was challenging to do while utilizing all the STL containers required because of certain containers such as the stack that I could not figure out how to shuffle, so instead I made a deque to store the cards in order first, and then shuffled and placed them into a stack. Taking cards from other players, placing them into the player's hand, checking for pairs, and throwing away pairs proved to be difficult. Debugging as best as I could for the actual playing process as described above took quite some time. Over 5 hours possibly. This project was done over the span of 4 days with about 4 hours working on it each day, while balancing other duties.

Description

A deque was used to create the cards and make the deck. The contents were shuffled in the deque and then pushed onto a stack. From the stack, the cards were popped off and erased from the stack onto lists. The lists were sorted. Used a loop to repeat the process of user selecting which card they would like to take and also to repeat the process of the computers to take cards from the corresponding player. Game ends when one player has the queen while the other players have empty hands. Used the .empty() function to determine this.

Sample Input/Output

This is how the deck looks:

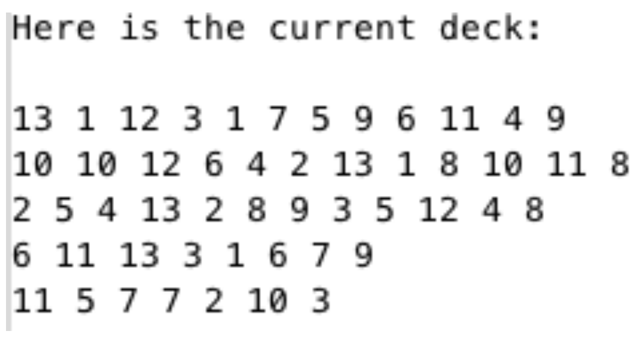


```
Here is the current deck:

1 1 1 1 2 2 2 2 3 3 3 3
4 4 4 4 5 5 5 5 6 6 6 6
7 7 7 7 8 8 8 8 9 9 9 9
10 10 10 10 11 11 11 11
12 12 12 13 13 13 13
```

There are 4 of each card
except for (12) which is a queen, there are only 3 queens in the deck.

Shuffling the deck:

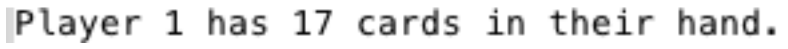


```
Here is the current deck:

13 1 12 3 1 7 5 9 6 11 4 9
10 10 12 6 4 2 13 1 8 10 11 8
2 5 4 13 2 8 9 3 5 12 4 8
6 11 13 3 1 6 7 9
11 5 7 7 2 10 3
```

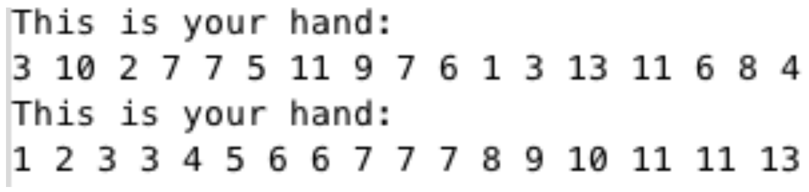
This is the shuffled deck.

Giving player's their share of the card, in this case, this is player 1 receiving cards. In this game, there are 3 players, so player 1 received 17 cards.



```
Player 1 has 17 cards in their hand.
```

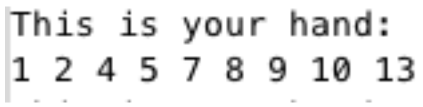
Outputting Player 1's hand:



```
This is your hand:
3 10 2 7 7 5 11 9 7 6 1 3 13 11 6 8 4
This is your hand:
1 2 3 3 4 5 6 6 7 7 7 8 9 10 11 11 13
```

As you can see, Player 1 received 17 cards that correspond to the last 17 cards in the stack. They are in backwards order from the stack because they have been popped off onto the list. The second hand is showing the player's hand in a sorted fashion.

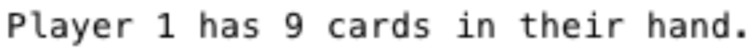
Player 1's hand after throwing away all pairs:



```
This is your hand:
1 2 4 5 7 8 9 10 13
```

The pair of 3's, 6's, 7's and 11's were thrown away from player 1's hand.

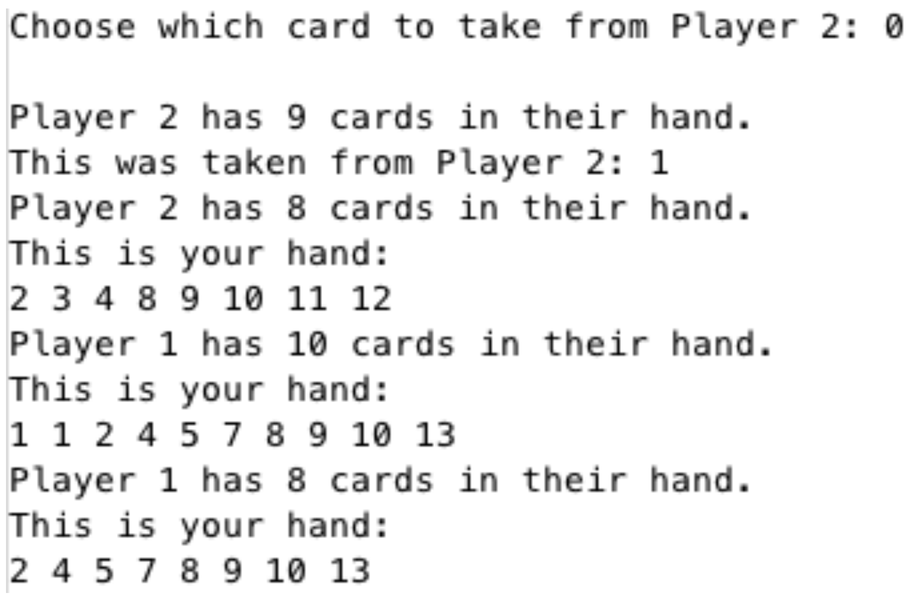
Total of player 1's hand after throwing away pairs:



```
Player 1 has 9 cards in their hand.
```

8 cards were thrown away, or 4 pairs. 17 - 8 = 9 cards left in player 1's hand. The same is done for computer players also.

Now the actual game begins, Player 1 is taking from Player 2:

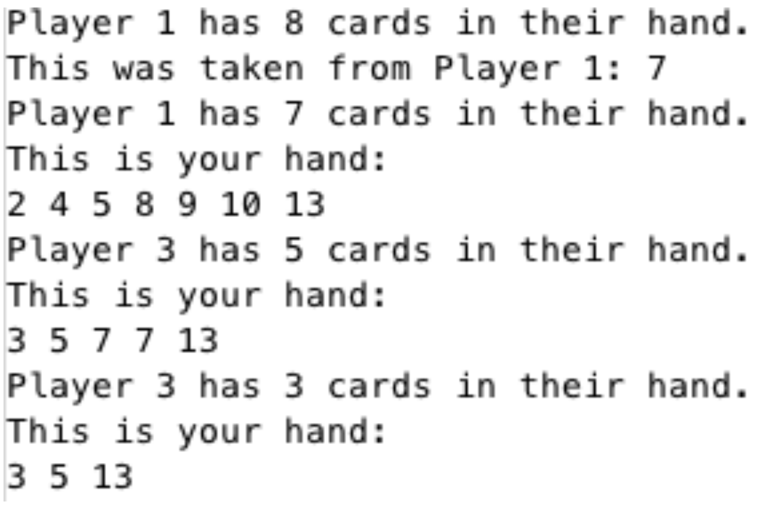


```
Choose which card to take from Player 2: 0

Player 2 has 9 cards in their hand.
This was taken from Player 2: 1
Player 2 has 8 cards in their hand.
This is your hand:
2 3 4 8 9 10 11 12
Player 1 has 10 cards in their hand.
This is your hand:
1 1 2 4 5 7 8 9 10 13
Player 1 has 8 cards in their hand.
This is your hand:
2 4 5 7 8 9 10 13
```

Player 1 inputs 0, which is the index for the first card in player 2's hand. Player 2 has 9 cards, and then the card 1 or Ace was taken from player 2, and now player 2's hand is 2, 3, 4, 8, 9, 10, 11, 12, totaling to be 8 cards in their hand. While player 1 has 1, 1, 2, 4, 5, 7, 8, 9, 10, 13, totaling to be 10 cards now instead of 9. After throwing away the pair of 1's, player 1 now only has 2, 4, 5, 7, 8, 9, 10, 13, totaling to be 8 cards in their hand.

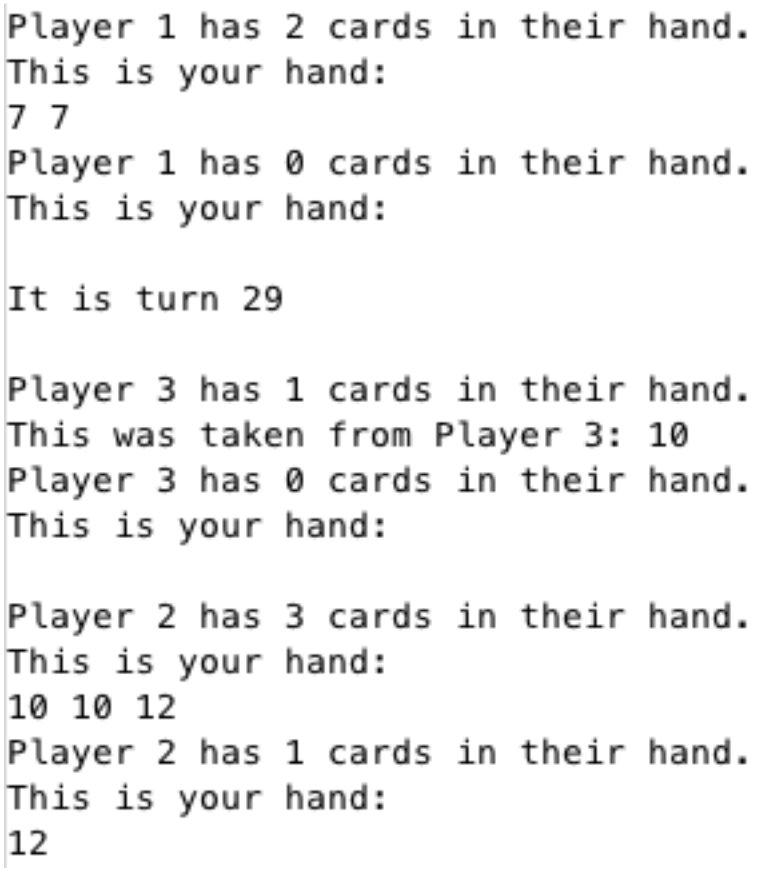
Demonstration of CPUs taking cards:



```
Player 1 has 8 cards in their hand.
This was taken from Player 1: 7
Player 1 has 7 cards in their hand.
This is your hand:
2 4 5 8 9 10 13
Player 3 has 5 cards in their hand.
This is your hand:
3 5 7 7 13
Player 3 has 3 cards in their hand.
This is your hand:
3 5 13
```

Player 1 has the same hand as the screenshot before, Player 3 which is computer takes the 7 in Player 1’s hand. Player 1 now has 2, 4, 5, 8, 9, 10, 13, totaling to be 7 cards now. Player 3’s hand is shown for demonstration purposes and has 3, 5, 7, 7, 13 after taking the card, totaling to be 5 cards. After throwing the pair of 7’s it is just 3, 5, 13, totaling to 3 cards.

Player 2 losing:



```
Player 1 has 2 cards in their hand.
This is your hand:
7 7
Player 1 has 0 cards in their hand.
This is your hand:

It is turn 29

Player 3 has 1 cards in their hand.
This was taken from Player 3: 10
Player 3 has 0 cards in their hand.
This is your hand:

Player 2 has 3 cards in their hand.
This is your hand:
10 10 12
Player 2 has 1 cards in their hand.
This is your hand:
12
```

Player 1 had a pair and threw it out leaving Player 1's hand empty meaning Player 1 is done. The same applies for Player 3 because Player 2 had no choice but to take it from Player 3, leaving Player 3 with an empty hand also. Player 2 is left with a pair, but is also left with the last remaining queen(12) and with no other Players left to take from, Player 2 loses.

Flowchart

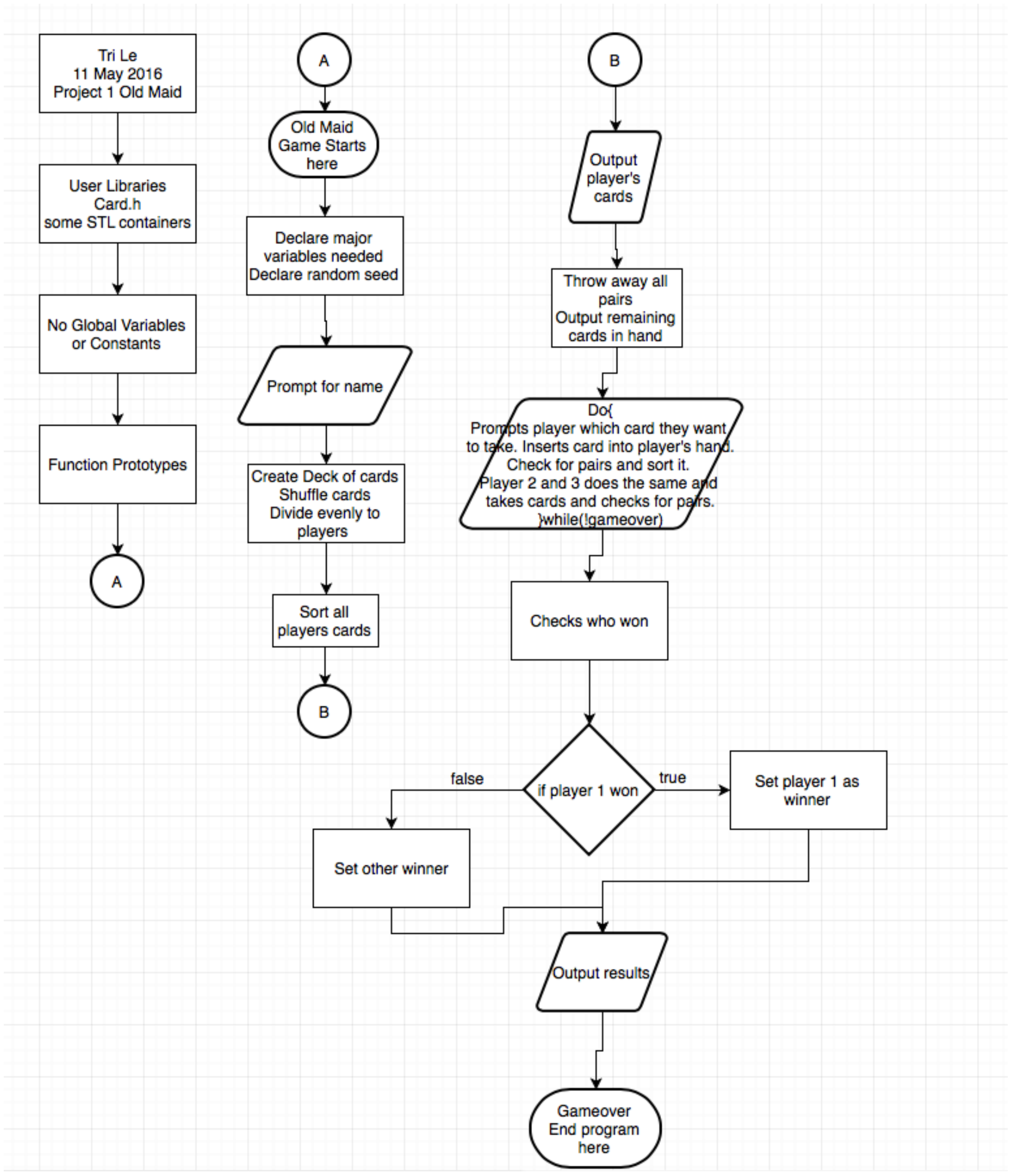


Image can also be found in the .zip file.

Pseudocode

```
Execution begins
Initialization

Enter name

Create deck
Shuffle the deck

Divide cards to players
Create hands for each player

Show the players hand
Throw away any pairs
Sort the hand again and show to player

Do the same as above for computers

Do{
        Player 1 chooses random card to take from Player 2
        Player 1 inserts into hand and checks for pairs
        Player 2 chooses random card to take from Player 3
        Player 2 inserts into hand and checks for pairs
        Player 3 chooses random card from Player 1
        Player 3 inserts into hand and checks for pairs
}while(Player1/2/3 hand ! empty);

Declare loser
Output results

Execution end
```

Variables

Type	Name	Description	Location
string	p1	Name of player 1	main, line 27
string	cpu2	Name of player 2 (cpu)	main, line 25
string	cpu3	Name of player 3 (cpu)	main, line 26
int	playerSz	Player count	main, line 40
int	counter	Used to determine who will receive one less card if playing in a game where there is an uneven split of cards	main, line 41
int	turn	keeps track of turns	main, line 86
int	whichCard	which card to take from player's hand	main, line 90
int	cardVal	card value that was taken from player's hand	main, line 91
int	curSize	current size of player's hand	hand.h, line 19
bool	p1win	checks if player 1 won	main, line 87
bool	p2win	checks if player 2 won	main, line 88
bool	p3win	checks if player 3 won	main, line 89
std::deque<int>	deck	initial unshuffled deck	main, line 30
std::stack<int>	newDeck	shuffled deck	main, line 38
std::list<int>	cardHand	list for player's hand	hand.h, line 18
std::map<string, string>	winOrder	map for win order of players	main, line 24

Concepts

std::deque<int> was used to create and store the initial deck
std::stack<int> was used to store the shuffled deck
std::list<int> was used as the player's hands after getting the contents from the stack

std::map<string, string> was used to display the player’s name and what place they were in at the end of the game
std::iterators were used to output the lists or to alter any contents in the list, also used to output the map
std::iterators can be found in hand.cpp

References

The Standard Template Library was used to create and hold the card deck
StackExchange was used to learn how to properly use iterators

*****The following code was built and complied on the Mac OS X operating system*****

Program

```
/*
 * File:   main.cpp
 * Author: Tri Le
 * CSC 17C Project 1 Old Maid Card Game
 * Created on May 7, 2016, 8:54 PM
 */

#include <cstdlib>
#include <iostream>
#include <ctime>
#include <map>
using namespace std;

#include "Hand.h"

void makeDeck(deque<int> &);                    // Make a deck
void showDeck(deque<int> &);                    // Print the deck
void shuffle(deque<int> &);                   // Shuffle the deck
void stackDeck(stack<int> &, deque<int>);        // Push deck onto a stack to divide into lists
later

int main(int argc, char** argv) {

   srand(time(0));
   map<string, string> winOrder;               // Used to store winners name in order
   string cpu2 = "Computer2";
   string cpu3 = "Computer3";
   string p1;
   cout << "Enter your name: ";
   cin >> p1;
   deque<int> deck;
   makeDeck(deck);
   int deckSize = deck.size();
   cout << "The deck size is " << deckSize << " after the removal of one QUEEN,"
           " which is represented by a (12)." << endl;
   showDeck(deck);
   shuffle(deck);
   showDeck(deck);
   stack<int> newDeck;
   stackDeck(newDeck, deck);
   int playerSz = 3;
   int counter = 0;
   Hand player1(newDeck, playerSz, deckSize, counter);
   player1.showHand();
   player1.sortHand();
   player1.showHand();
   cout << endl;
   Hand player2(newDeck, playerSz, deckSize, counter);
   player2.showHand();
   player2.sortHand();
   player2.showHand();
   cout << endl;
   Hand player3(newDeck, playerSz, deckSize, counter);
   player3.showHand();
   player3.sortHand();
   player3.showHand();
   cout << endl;

   player1.setCurSize();
   player2.setCurSize();
   player3.setCurSize();
   cout << "Player 1 has " << player1.getCurSize() << " cards in their hand."
       << endl;
   cout << "Player 2 has " << player2.getCurSize() << " cards in their hand."
       << endl;
   cout << "Player 3 has " << player3.getCurSize() << " cards in their hand."
       << endl;

   player1.throwPairs();
```

```
player1.showHand();
player2.throwPairs();
player2.showHand();
player3.throwPairs();
player3.showHand();

player1.setCurSize();
player2.setCurSize();
player3.setCurSize();

cout << "Player 1 has " << player1.getCurSize() << " cards in their hand."
    << endl;
cout << "Player 2 has " << player2.getCurSize() << " cards in their hand."
    << endl;
cout << "Player 3 has " << player3.getCurSize() << " cards in their hand."
    << endl;

int turn = 1;                   // Turn counter
bool p1win = false;                 // Check for P1 Win
bool p2win = false;                 // Check for P2 Win
bool p3win = false;                 // Check for P3 Win
int whichCard = 0;                  // Make sure this number doesn't go out of bounds
int cardVal = 0;                  // Used to store what card was taken

do{
// Player 1 takes from Player 2's hand
cout << "It is turn " << turn << endl;
cout << "Choose which card to take from Player 2: ";
do{
   cin >> whichCard;
}while(whichCard > player2.getCurSize() - 1);
cardVal = player2.takeCard(whichCard);
cout << endl;
cout << "Player 2 has " << player2.getCurSize() << " cards in their hand."
    << endl;
cout << "This was taken from Player 2: " << cardVal << endl;
player2.setCurSize();
cout << "Player 2 has " << player2.getCurSize() << " cards in their hand."
    << endl;
player2.showHand();
player1.insertCard(cardVal);
player1.setCurSize();
cout << "Player 1 has " << player1.getCurSize() << " cards in their hand."
```

```
    << endl;
player1.showHand();
player1.throwPairs();
player1.setCurSize();
cout << "Player 1 has " << player1.getCurSize() << " cards in their hand."
    << endl;
player1.showHand();
player1.checkWin(p1win);
turn++;
// Player 2 takes from Player 3's hand
cout << "It is turn " << turn << endl;
whichCard = player3.cpuTake(player3.getCurSize());
cardVal = player3.takeCard(whichCard);
cout << endl;
cout << "Player 3 has " << player3.getCurSize() << " cards in their hand."
    << endl;
cout << "This was taken from Player 3: " << cardVal << endl;
player3.setCurSize();
cout << "Player 3 has " << player3.getCurSize() << " cards in their hand."
    << endl;
player3.showHand();
player2.insertCard(cardVal);
player2.setCurSize();
cout << "Player 2 has " << player2.getCurSize() << " cards in their hand."
    << endl;
player2.showHand();
player2.throwPairs();
player2.setCurSize();
cout << "Player 2 has " << player2.getCurSize() << " cards in their hand."
    << endl;
player2.showHand();
player2.checkWin(p2win);
turn++;
//Player 3 takes from Player 1's hand
cout << "It is turn " << turn << endl;
whichCard = player1.cpuTake(player1.getCurSize());
cardVal = player1.takeCard(whichCard);
cout << endl;
cout << "Player 1 has " << player1.getCurSize() << " cards in their hand."
    << endl;
cout << "This was taken from Player 1: " << cardVal << endl;
player1.setCurSize();
cout << "Player 1 has " << player1.getCurSize() << " cards in their hand."
```

```
        << endl;
    player1.showHand();
    player3.insertCard(cardVal);
    player3.setCurSize();
    cout << "Player 3 has " << player3.getCurSize() << " cards in their hand."
        << endl;
    player3.showHand();
    player3.throwPairs();
    player3.setCurSize();
    cout << "Player 3 has " << player3.getCurSize() << " cards in their hand."
        << endl;
    player3.showHand();
    player3.checkWin(p3win);
    turn++;
    // Outputting winner
    if(player1.getCurSize() == 0){
      winOrder[p1] = "First";
      winOrder[cpu2] = "Second";
      winOrder[cpu3] = "Third";
      for(map<string, string>::iterator i = winOrder.begin(); i != winOrder.end(); i++){
        cout << i->first << " " << i->second << endl;
      }
    }
    else if(player2.getCurSize() == 0){
      winOrder[cpu2] = "First";
      winOrder[p1] = "Second";
      winOrder[cpu3] = "Third";
      for(map<string, string>::iterator i = winOrder.begin(); i != winOrder.end(); i++){
        cout << i->first << " " << i->second << endl;
      }
    }
    else if(player3.getCurSize() == 0){
      winOrder[cpu3] = "First";
      winOrder[p1] = "Second";
      winOrder[cpu2] = "Third";
      for(map<string, string>::iterator i = winOrder.begin(); i != winOrder.end(); i++){
        cout << i->first << " " << i->second << endl;
      }
    }
    }while(p1win == false || p2win == false || p3win == false);

    return 0;
}
```

```
void makeDeck(deque<int> &deck){
   bool flag = false;
   for(int i=1;i<14;i++){
      if(i != 12){
      for(int j=0;j<4;j++){
         deck.push_back(i);
      }
      }
      if(i==12)
         flag = true;
      if(flag == true){
         for(int q=0;q<3;q++){
            deck.push_back(i);
         }
         flag = false;
      }
   }
}

void showDeck(deque<int> &deck){                  // Shows all the cards in the deck for testing
purposes
   cout << "Here is the current deck: " << endl << endl;
   for(int i=0;i<deck.size();i++){
      cout << deck[i] << " ";
      if(i==11 || i==23 || i==35 || i==43)
         cout << endl;
   }
   cout << endl << endl;
}

void shuffle(deque<int> &deck){                  // Shuffles all the cards in the deck
   for(int first = 0; first < deck.size(); first++){
      // Create an int called second and set it equal to the random operator
      int second = (rand() + time(0)) % deck.size();
      int temp = deck[first];
      deck[first] = deck[second];
      deck[second] = temp;
   }
}

void stackDeck(stack<int> &newDeck, deque<int> deck){
   for(int i=0;i<51;i++){
        newDeck.push(deck[i]);
    }
}

/*
 * File:   Hand.h
 * Author: Tri Le
 * CSC 17C Project 1 Old Maid Card Game
 * Created on May 8, 2016, 8:42 PM
 */

#ifndef HAND_H
#defineHAND_H

#include <list>
#include <stack>
#include <cstdlib>
#include <ctime>

class Hand {
private:
    std::list<int> cardHand;                            // Any given player's hand of cards
    int curSize;                                        // Current size of the player's hand
public:
    Hand();                                             // Default constructor sets empty list
    Hand(std::stack<int> &, int, int, int &);           // Constructor takes in the shuffled deck and
splits it equally
                                                        // among X amount of players and checks who is last to
receive cards
    void showHand();                                    // Shows the player's hand
    void sortHand();                                    // Sorts the player's hand
    void throwPairs();                                  // Removes pairs from the player's hand
    void setCurSize();                                  // Sets the player's current hand size
    int getCurSize() { return curSize; }                // Return the player's current hand size
    int takeCard(int);                                  // Take a card from another player
    void insertCard(int);                               // Places taken card into player's hand
    int cpuTake(int);                                   // Chooses which card the CPU will take
    void checkWin(bool &);                              // Checks for a win
};

#endif  /* HAND_H */
```

```
/*
 * File:   Hand.cpp
 * Author: Tri Le
 * CSC 17C Project 1 Old Maid Card Game
 * Created on May 8, 2016, 8:42 PM
 */

#include <cstdlib>
#include <iostream>
#include "Hand.h"

Hand::Hand() {
}
// Constructor makes deck in order
Hand::Hand(std::stack<int> &cards, int players, int limit, int &counter){
    int split;
    if(limit % players == 0){
        split = limit / players;
        for(int i=0; i<split; i++){
            cardHand.push_back(cards.top());
            cards.pop();
        }
    }
    else{
        split = (limit / players) + 1;
        if(counter < players - 1){
            counter++;
            for(int i=0; i<split; i++){
                cardHand.push_back(cards.top());
                cards.pop();
            }
        }
        else{
            for(int i=0; i<split - 1; i++){
                cardHand.push_back(cards.top());
                cards.pop();
            }
        }
    }
}
// Outputs hand
void Hand::showHand(){
    std::cout << "This is your hand: " << std::endl;
    std::list<int>::iterator i;
    for(i = cardHand.begin(); i != cardHand.end(); i++){
        std::cout << *i << " ";
    }
    std::cout << std::endl;
}
// Sorts hand
void Hand::sortHand(){
    cardHand.sort();
}
// Throws away any pairs in the hand
void Hand::throwPairs(){
    for (std::list<int>::iterator i = cardHand.begin(); i != cardHand.end();){
        std::list<int>::iterator n = i;
        n++;
        if (n == cardHand.end())
            break;
        if (*i == *n){
            i = cardHand.erase(i);
            i = cardHand.erase(i);
        }
        else
            i++;
    }
}

void Hand::setCurSize(){
    curSize = cardHand.size();
}
// Take selected card from other player hand
int Hand::takeCard(int pos){
    int temp;
    if (pos < cardHand.size()){
        std::list<int>::iterator i = cardHand.begin();
        std::advance(i, pos);                    // 'i' points to the element at index 'N'
        temp = *i;
        i = cardHand.erase(i);
    }
    return temp;
}
// Insert card into player's hand
void Hand::insertCard(int cardVal){
    cardHand.push_back(cardVal);
    cardHand.sort();
}
// Computer randomly choosing index to take from other player's hand
int Hand::cpuTake(int size){
    int temp;
    temp = rand() % size;
    return temp;
}
// Check for a win
void Hand::checkWin(bool &win){
    if(cardHand.empty()){
        win = true;
    }
    else{
        win = false;
    }
}
```