

INN371 – Data Structures and Algorithms

Assignment 1

Contract Bridge Opening Bids

Due Date: 11:59 pm – Sunday 20th April, 2014

Weighting: 20%

Individual Assignment

Introduction

Contract Bridge is one of the world's most popular card games and is played world-wide in clubs, in tournaments, online and socially. It is a **trick-taking** game using a standard 52 card deck. The cards are in four **suits** – Spades, Hearts, Diamond and Clubs – and each card has a **rank**. The ranks from lowest to highest in each suit are 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King and finally Ace.

Many trick-taking games, including Contract Bridge, designate one of the suits to be a **trump suit**. The decision about which suit will be the trump suit is part of each hand and is known as the **auction**. On each trick, where each player plays one card from their hand, the players must play a card of the same suit as the card that was lead if they have one in their hand. If they **do not** have a card in the suit which was lead they may play a card from any other suit. If they play a card from the trump suit, that card beats the cards from the original suit that was played.

For example, if Spades were trumps and the first player lead the Ten of Diamonds, the second player followed suit and played the Ace of Diamonds, the third player who does not have any Diamonds may play a Spade. If they play the Two of Spades, the 2S is now winning the trick and can only be beaten if the fourth player also does not have any Diamonds and can play a higher trump card i.e. a Spade.

Based on the cards received, the players may decide that they will not make any suit the trump suit. In this case the hand will be played as **no trumps** and all cards have their natural ranks with the highest card on each trick being the winner.

Players work in partnerships and sit on opposite sides of a four-sided table. The players are designated, **in order**, North, East, South and West. North and South play together as do East and West.

The game starts with North designated as the initial dealer. The game consists of several deals or hands each of which has distinct phases:

- The Deal – the dealer deals the cards **clockwise** one at a time, starting with the player on their **left** and continuing until all cards are dealt, so that each player has 13 cards. The deal rotates **clockwise** after each hand.
- The Auction – the players make **bids** (see below for more information) indicating the strength and shape (number of cards per suit) of their hands until a contract is reached. The dealer is the first player to decide if they can make the **opening bid**. Bids continue in a clockwise direction until three players in succession each make a **Pass** bid. The final contract states which suit will be the trump suit and how many tricks the partnership winning the auction has to win in order to score points.
- The Play – The opening lead is made by the player on the left of the player who bid the final trump suit first. For example, if South opened the bidding 1 Diamonds, but North was the first player to bid Hearts and their bid was 1 Hearts with the final contract of 4 Hearts being made by South then East (to the left of North) would make the opening lead. Each subsequent lead is made by the winner of the previous trick.
- Scoring – Points are awarded to the partnerships based on whether or not they won as many or more tricks as they said they would in the auction.

Bidding

For the purposes of bidding, the suits in Contract Bridge are ordered from low to high as follows: Clubs, Diamonds, Hearts and Spades. Clubs and Diamonds are named **minor suits**. Hearts and Spades are named **major suits**. In the bidding, no trumps is ranked higher than Spades.

The lowest bid that can be made is 1 Club, followed by 1 Diamonds, 1 Hearts, 1 Spades, 1 No Trumps, 2 Clubs, 2 Diamonds and so on. The highest bid that can be made is 7 No Trumps.

Task

This assignment will only involve setting up the deck of cards, shuffling and dealing the cards to the players and deciding on who will make the **opening bid** for each hand.

Your program will do the following:

1. Create a standard deck of 52 playing cards with cards ranked 2, 3, 4, 5, 6, 7, 8, 9, T (ten), J (jack), Q (queen), K (king) and A (ace), in four suits C (clubs), D (diamonds), H (hearts) and S (spades).
2. North is the dealer for the first hand.
3. For four hands in succession
 - a. Shuffle the deck into a random ordering
 - b. Deal thirteen (13) cards to the four (4) players. The first hand is dealt by North, so the first card is received by East. Cards are dealt in player order

(North, East, South, West). All four players receive their first card, before receiving their second card and so on.

- c. Based on the bidding logic outlined below decide which player makes the opening bid (other than Pass).
- d. Output the four hands in order starting with the dealer of the current hand. The cards for each hand should be displayed in descending suit order and in descending rank order. See the screenshot below for the required output format.
- e. Output the opening bid and who has made the bid. See the screenshot below for the required output format.
- f. Pass the deal for the next hand on to the next player in clockwise order (N, E, S, W).

```

D:\Users\Malcolm\Dropbox\W...
NORTH
Spades : 4S
Hearts : QH TH 4H 3H
Diamonds: 4D
Clubs : AC KC TC 8C 6C 5C 3C

EAST
Spades : TS 8S
Hearts : 9H 8H 6H 5H
Diamonds: 7D 6D 5D 2D
Clubs : 9C 7C 2C

SOUTH
Spades : AS 7S 3S 2S
Hearts : AH KH JH 2H
Diamonds: KD JD TD 3D
Clubs : JC

WEST
Spades : KS QS JS 9S 6S 5S
Hearts : 7H
Diamonds: AD QD 9D 8D
Clubs : QC 4C

Opening Bid is 3C made by NORTH

=====

EAST
Spades : AS JS TS 8S 6S 4S
Hearts : AH
Diamonds: 7D 6D 5D
Clubs : KC TC 5C

SOUTH
Spades : QS 7S 5S 3S
Hearts : JH 9H 6H 2H
Diamonds: JD TD 4D 2D
Clubs : 3C

WEST
Spades : KS 2S
Hearts : 5H
Diamonds: AD KD 8D 3D
Clubs : AC 9C 8C 7C 6C 4C

NORTH
Spades : 9S
Hearts : KH QH TH 8H 7H 4H 3H
Diamonds: QD 9D
Clubs : QC JC 2C

Opening Bid is 1S made by EAST

=====

```

```

D:\Users\Malcolm\Dropbox\W...
SOUTH
Spades : 5S 4S 3S
Hearts : AH 3H
Diamonds: AD TD 7D 2D
Clubs : AC TC 3C 2C

WEST
Spades : AS KS QS 2S
Hearts : 5H 2H
Diamonds: JD 9D
Clubs : JC 7C 6C 5C 4C

NORTH
Spades : JS TS 8S 7S
Hearts : KH QH JH 7H
Diamonds: QD 8D 4D
Clubs : 9C 8C

EAST
Spades : 9S 6S
Hearts : TH 9H 8H 6H 4H
Diamonds: KD 6D 5D 3D
Clubs : KC QC

All hands PASS

=====

WEST
Spades : AS TS 5S 2S
Hearts : AH KH 9H 2H
Diamonds: AD 9D 7D
Clubs : KC 7C

NORTH
Spades : KS 8S 7S 4S
Hearts : JH 8H 7H
Diamonds: KD QD
Clubs : AC JC 8C 4C

EAST
Spades : JS
Hearts : QH TH 5H 4H
Diamonds: TD 6D 4D 3D
Clubs : 6C 5C 3C 2C

SOUTH
Spades : QS 9S 6S 3S
Hearts : 6H 3H
Diamonds: JD 8D 5D 2D
Clubs : QC TC 9C

Opening Bid is 1D made by WEST

=====

```

Example of Expected Output

Bidding Logic

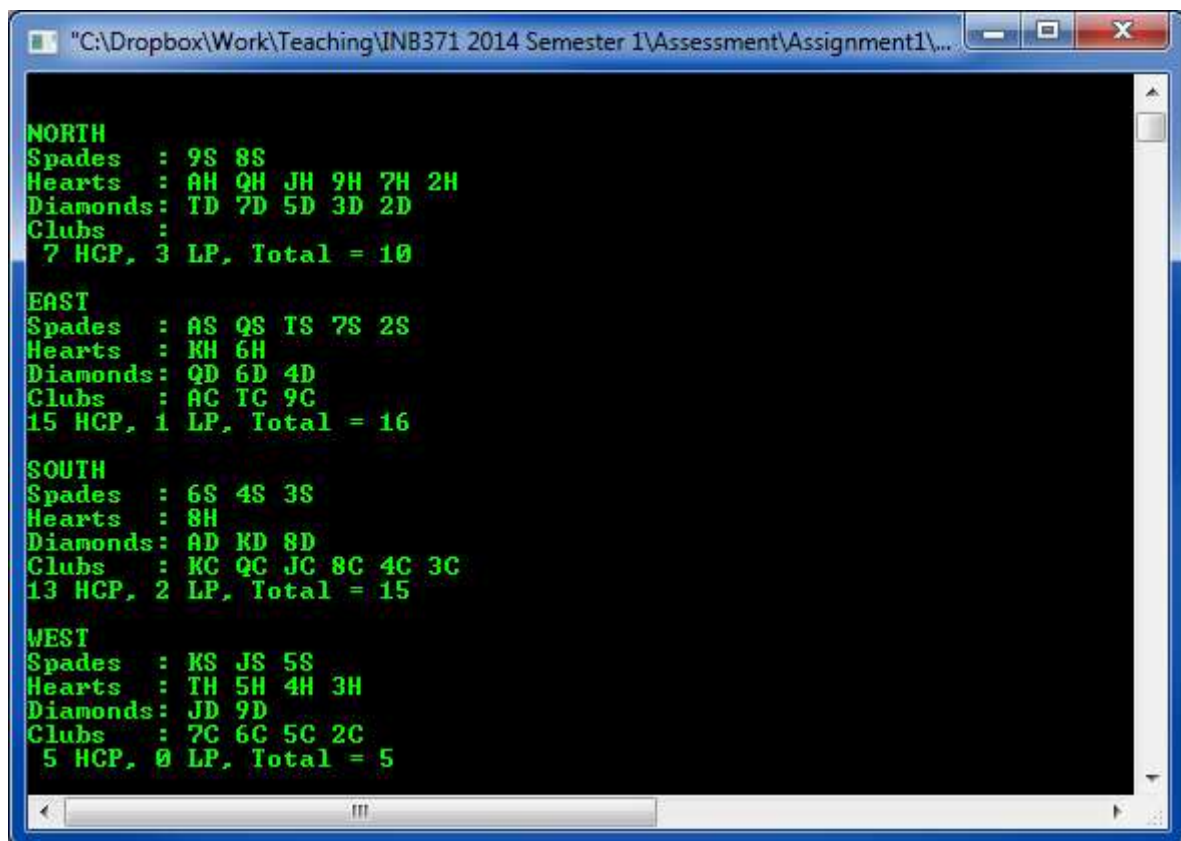
To decide on a bid a player must evaluate the **strength** and **shape** of their hand. Once these properties have been determined, the **bidding logic** can be applied.

Strength

The strength of the hand is calculated by adding the number of **high card points** and the number of **length points** in the hand.

- High Card Points – Calculated as the total number of points in the hand where Ace = 4, King = 3, Queen = 2, Jack = 1
- Length Points – 1 point for every card after the 4th card in a suit

An example deal is shown below with the high card points, length points and total points displayed below the four suits.



```
"C:\Dropbox\Work\Teaching\INB371 2014 Semester 1\Assessment\Assignment1\...  
NORTH  
Spades : 9S 8S  
Hearts : AH QH JH 9H 7H 2H  
Diamonds: 1D 7D 5D 3D 2D  
Clubs :  
7 HCP, 3 LP, Total = 10  
EAST  
Spades : AS QS TS 7S 2S  
Hearts : KH 6H  
Diamonds: QD 6D 4D  
Clubs : AC TC 9C  
15 HCP, 1 LP, Total = 16  
SOUTH  
Spades : 6S 4S 3S  
Hearts : 8H  
Diamonds: AD KD 8D  
Clubs : KC QC JC 8C 4C 3C  
13 HCP, 2 LP, Total = 15  
WEST  
Spades : KS JS 5S  
Hearts : TH 5H 4H 3H  
Diamonds: JD 9D  
Clubs : 7C 6C 5C 2C  
5 HCP, 0 LP, Total = 5
```

Shape

The shape of a hand is either **balanced** or **unbalanced**.

- **Balanced** – A balanced hand is a hand where **no suit** has just **zero or one card or five or more cards** and has **at most one suit** with two cards. **The only distribution of the lengths of the suits for balanced hands are 4-4-3-2 or 4-3-3-3.**
- **Unbalanced** – An unbalanced hand is one which has any other distribution of suit lengths i.e. it is not balanced.

Bidding Logic

You generally need at least 13 points to open the bidding except if the hand is unbalanced and has a 6-card, 7-card or 8-card suit.

Unbalanced Hand

- 0 to 12 points – **Pre-emptive bids**
 - With a 6-card suit, bid 2 of the suit (**except Clubs** – if the 6-card suit is clubs, PASS)
 - **If there are two 6-card suits, bid the highest of these suits**
 - With a 7-card suit, bid 3 of the suit
 - With an 8-card suit, bid 4 of the suit
 - Otherwise PASS
- 13 to 21 points – **1-level suit opening**
 - Bid 1 of the longest suit
 - When two suits have equal length ≥ 5 , bid the higher suit (e.g. Spades rather than Hearts)
 - If the longest suit has length 4 and there is more than one suit, bid the lowest of these suits
- 22 or more points
 - Bid 2C

Balanced Hand

- 0 to 12 points – PASS
- 13 to 14 points – Bid a minor suit
 - With equal length in the minor suits
 - Bid 1D if the suits are of length 4
 - Bid 1C if the suits are of length 3
 - Otherwise bid 1 of the longest minor suit (Diamonds and Clubs)
- 15 to 17 points – Bid 1 NT (No Trump)
- 18 to 19 points – Same as for 13 to 14 points
- 20 to 21 points – Bid 2 NT (No Trump)
- 22 or more points – Bid 2C

Design Decisions

All classes **MUST** be defined separately with each class having a definition or header file (.h) and a separate implementation file (.cpp).

The following design decisions may aid in your implementation. It is strongly suggested that you follow these guidelines.

The solution that produces the above output was built by creating the following classes:

- **Random** – Produces random integers
- **Card** – Encapsulates a rank (2 to Ace) and suit (Diamonds, Clubs, Hearts, Spades)
- **Deck** – Encapsulates a collection of 52 individual **Card** objects, keeps track of the number of **cardsDealt**
- **Hand** – Encapsulates 13 **Card** objects that are dealt from the **Deck** of **Card** and information about the strength and shape of the hand from the cards within. Each card dealt to the hand can be placed into a container for each suit. The decision on which container type to use should be made considering that the output for each suit is ordered.
- **Game** – Encapsulates four **Hand** objects and the position of the dealer of the current deal

The Random Class

This class is based on the exercises included in Workshop 3. To ensure that different random values are generated on successive runs, the private **randomise()** method must be called by the constructor.

The Card Class

This class is based on the exercises included in Workshops 3 and 4. The header/interface file for this class, should also include definitions for the enumerations **Rank** and **Suit**.

Card()	No argument constructor
Card(Rank, Suit)	Constructor that sets the rank and suit for a Card object
Card(string)	Constructor that sets the rank and suit for a Card object from an input string
~Card()	Destructor – does nothing as no objects are created dynamically by the constructor of this class
Rank getRank()	Accessor for the Rank instance variable
Suit getSuit()	Accessor for the Suit instance variable
bool operator()(Card*, Card*)	Overloads the function operator to provide a comparison which sets an ordering between two Card objects.
friend ostream& operator<<(ostream&, Card&)	Puts a string representation of this Card on the output stream

The Deck Class

This class is based on the exercises included in Workshops 3 and 4.

Deck()	No Argument constructor – Creates a dynamically allocated array of Card* objects and initialises them. Initialises cardsDealt to 0. Note that the dynamically allocated array should contain <i>pointers</i> to Card to minimise the overhead of copying during the shuffling procedure. See Lecture 5, for an example of how this is done.
~Deck()	Destructor – deletes the contents of the array of Card*
void reset()	Sets cardsDealt to 0.
Card* dealNextCard()	Returns a pointer to the next Card object from the deck, increments cardsDealt
void shuffle()	Shuffles the cards in the deck. This can be achieved by selecting two random indexes in the array and swapping the references around, putting the cards out of their original order.
friend ostream& operator<<(ostream&, Deck&)	Puts a string representation of the Card objects in the deck to the screen. This method is useful during testing to ensure that shuffling is being achieved.
friend istream& operator>>(istream&, Deck&)	Reads 52 strings from an input stream, where each string represents a card e.g. 2C, AD, JH, constructing a Card* and assigning them into the array of Card*

The Hand Class

The **Hand** class is responsible for, among other things, storing the cards dealt to a particular player.

The **Hand** class is also responsible for determining the bid that the hand can make using the bidding logic described above. The class could encapsulate:

- A group of four collections of pointer to **Card**, one for each suit.
- The class will require private instance variables to record the number of points (strength of hand) and the shape of the hand. It may also have other private functions that help to determine the bid.

Hand()	No arguments constructor, initialises all suit collections
~Hand()	Destructor – deletes each Card* in the Hand
void clear()	Clears all of the suit collections, resets any instance variables to their initial values.
void addCard(Card*)	Adds a Card* to the Hand placing the card with other cards of the same suit
string makeBid()	Responsible for implementing the bidding logic and for deciding on the bid to be made for this hand. That bid could be PASS. Returns the bid as a string.
friend ostream& operator<<(ostream& out, Hand& hand)	Puts a string representation of a Hand on the output stream

The Game Class

The header file for this class should also provide an enumeration type **Position** for **NORTH**, **EAST**, **SOUTH** and **WEST**.

The **Game** class is responsible for providing the main functionality of the game. It should hold the **Deck** object for the game and should hold an array of **Hand** and should keep track of which **Position** is currently the **dealer**. It should also store the opening bid and who made the bid.

Game()	No args constructor, initialise the dynamic array of Hand , sets dealer to NORTH
~Game()	Destructor responsible for discarding the array
void setup(bool fromFile)	If input is NOT to be read from a file, shuffle the deck . Call the reset method on the deck . Also clears each Hand .
void deal()	Deals the cards from the deck clockwise to each player in turn starting with the player on the dealer's left.
void auction()	Starting with the dealer, calls makeBid() to see what bid is made. If a bid other than Pass is received from the current Hand , the auction can stop.
void nextDealer()	Increments the dealer Position to the next player.
friend ostream& operator<<(ostream& out, Game& game)	Puts a string representation of the game onto the output stream.
friend istream& operator>>(istream& in, Game& game)	Passes the input stream onto the deck so that the cards can be read from a text file.

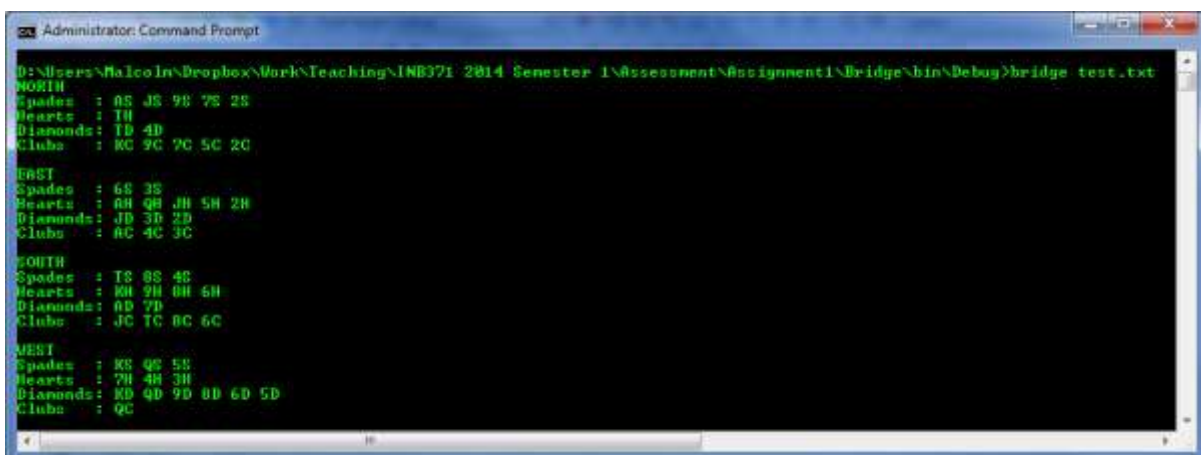
The Bridge Program

The program should:

- Declare a **Game**
- **setup** the **Game** instance
- Deal the cards
- Conduct the bidding auction
- Output the information for the current deal in the format given in the sample runs

A completed file has been provided (**bridge.cpp**) which includes the ability to read a predetermined **Deck** into the program. Some text files will be provided so that you can check the reliability of your program. For example, the file **test.txt** will produce output when run from the command line as **bridge test.txt**. The program will terminate with an error if the file given as a command line argument does not exist.

If a Command Line Argument is not given, the program should produce output from a deck which has been shuffled randomly.



```
Administrator: Command Prompt
D:\Users\Malco In\Dropbox\Work\Teaching\INB371 2014 Semester 1\Assessment\Assignment1\Bridge\bin\Debug>bridge test.txt
NORTH
Spades : AS JS 9S 7S 2S
Hearts : TH
Diamonds: TD 4D
Clubs : KC 9C 7C 5C 2C

EAST
Spades : 6S 3S
Hearts : 6H 4H JH 5H 2H
Diamonds: JD 3D 2D
Clubs : AC 4C 3C

SOUTH
Spades : TS 8S 4S
Hearts : KH 9H 8H 6H
Diamonds: 10D 7D
Clubs : JC TC 8C 6C

WEST
Spades : KS QS 5S
Hearts : 7H 4H 3H
Diamonds: KD QD 9D 8D 6D 5D
Clubs : QC
```