CARD class
CARD( const E_RANK, const E_SUIT )
Description: The constructor for the card class.
Parameters: 1) The rank of the card of type E_RANK
            2) The suit of the card of type E_SUIT
Return value: None
char* getSuit()
Description: Gets the suit of the card.
Parameters: None.
Return Value: The suit of the card of type char*.
char* getRank()
Description: Gets the rank of the card.
Parameters: None.
Return Value: The rank of the card of type char*.
char* path()
Description: Gets the path of the picture of the current card.
Parameters: None.
Return Value: The path of the picture of the card relative to the current directory.
E_RANK myrank() const;
Description: Gets the rank of the card.
Parameters: None.
Return Value: The rank of the card of type E_RANK.
E_SUIT mysuit() const;
Description: Gets the suit of the card.
Parameters: None.
Return Value: The suit of the card of type E_SUIT.
**DECK class**

DECK()

Description: The constructor of the DECK class.
Parameters: None.
Return Value: None.

~DECK()

Description: The destructor of the DECK class
Parameters: None.
Return Value: None.

CARD* getCARD()

Description: Gets the next CARD from the deck
Parameters: None.
Return Value: The next CARD from the deck.

**GAMEPLAY class**

The gameplay class is not really a data structure as it just contains functions to control the flow of the game. Such functions include start() to send the initial force_bet packets to each player to post the blinds, player_deal() to send card packets to each player, table_deal() to send card packets for display on the board, winner() to call the hand evaluator functions, etc.

**hand class**

hand()

Description: The constructor for the hand class.
Parameters: None.
Return Value: None.

void straight()
void flush()
void nofakind()

Description: Special functions to evaluate special hand types.
Parameters: None.
Return Value: None.
void evaluate()
    Description: Determines what type of hand (straight, flush, etc) this hand is using the hand evaluator map.
    Parameters: None.
    Return Value: None.

E_TYPE getbesthand() const
    Description: Gets the type of this hand (straight, flush, etc).
    Parameters: None.
    Return Value: The type of this hand of type E_TYPE.

list<CARD*>* gethand()
    Description: Gets the best 5 cards in the hand.
    Parameters: None.
    Return Value: A pointer to a list of pointers to CARDs.

list<CARD*>* getoriginal()
    Description: Gets all 7 cards of a player’s hand (community + pocket).
    Parameters: None.
    Return Value: A pointer to a list of pointers to CARDs.

**PACKET class**
Not really a data structure. Just holds other data structures and processes them for transmission over the internet.

Structure:
    Byte 1: Packet type
    Byte 2: Sender
    Bytes 3-7: Size of the data portion of the packet
    Bytes 8-end: Data portion of the packet

**PLAYER class**
Not really a unique data structure. Just contains a HAND, the player’s name, type (SERVERPLAYER or CLIENTPLAYER) and cash and bet amounts and functions to access them.

**Form1 class**
Contains the controls of the GUI such as the table, picture controls to display the cards, buttons and slidebars for the bet controls, and also functions to send data through the socket when events are triggered from the controls (i.e. the user clicks a button).
**Form_setup class**

Basically acts as the server for the SERVERCLIENT and just establishes the connection for the CLIENT. There are controls such as textboxes to input the player’s name and radio buttons to select client or server. It includes functions to establish the connection and also to handle the different types of packets as they come in.

**Standard Data Structures Used**

All our key data structures are from the STL classes. The following are references to APIs for these structures:

1) **Maps**

2) **Lists**
   - Partial list: Ford & Topp pgs. 287,291,295,303

3) **Vectors**
   - Partial list: Ford & Topp pgs. 201-202