# Texas Hold'em Poker

By Sebastian Hall


## Introduction

This is the most well known version of poker that originated in the early twentieth century. The game allows for between two to six players to play. Each player aims to get the best five card combination of the seven available with their two hole cards and the five community cards. The objective of the game is to win as many chips as possible. This is the most commonly played game in casinos in America as well as the most commonly portrayed version of poker in popular culture.

# Rules

The players each begin with 100 chips each round they can wager those chips on their hand of cards or bow out. Each player is dealt two hole cards per round that are exclusive to their hand and there are four betting rounds. After the preliminary betting round three community cards are revealed that can be used for hands for anyone at the table and one additional card for the next two rounds. After the fifth and final card is drawn there is a fourth round of betting before hands are shown and compared. After a round has concluded the player can choose to continue playing or quit if they have chips remaining or the game ends with them losing if they have run out of chips.

# The Game

The game consists of a command line interface where the user is prompted to select between 1 and 5 AI players to play with. After that they enter their name of choice and are then placed into the first round with their selected number of opponents. Each round consists of five phases, four of which are the four betting phases and the fifth is determining the winner, called the pre flop, flop, turn, river, and showdown. In each phase the player has the option to call to the minimum bet, check if they already have, raise above the minimum bet by 10 or 20 chips, bet all of their chips, or fold for that round. These options are available to the user through a

numbered menu listing their choices. If the player chooses to fold or go all in the remaining phases are gone through quickly and the results are printed quickly since it is only AI playing. Each phase is played starting with the player to the left of the dealer and moving left which would be clockwise if playing in a circle and the phase continues until every player has made a move at least once and all players have called to the minimum bet or folded. The community cards are listed at the start of each phase and a turn any player makes is listed as a short one line description. The human player can see their cards, remaining chips, and the amount in the pot at each of their turns. After the player wins, loses, or decides to quit then the score ranking is shown. It lists players names sorted by the number of chips they ended with.

## STL Containers

- Set
    - A set is used as the container for each players hand of cards. It is useful in this regard because, when printed out, it will have the elements sorted according the card sorting order (default pair sorting order, rank first and then suit) and have any N cards of a kind near each other so that they are easier to spot on a somewhat difficult to use command line interface.
- Map

- A map is used in the player class to store string to string key value pairs describing the player. This player info is used whenever a players turn is output on the screen. The possible fields in the map include always their name and optionally special statuses like if they are the dealer, small blind, big blind, or any valid combination of those.

- Queue
  - The community cards are implemented using a queue because the input – output order made is useful to do so. Each time a card is drawn from the deck it is placed at the end of the queue and it is never sorted like other containers so the original card drawing order remains intact every time a new card is added and it has the oldest cards at the front of the queue.
- Priority Queue
  - The priority queue is used for storing the results list of players. Each time a player is knocked out of the game they are removed from the player list and added to the results priority_queue and once the player quits the remaining players are added as well. After overloading the comparison operator for players this was an easy choice for results because the container is sorted with the highest priority, meaning highest chip count, players being at the front of the queue. That works well for scoring results.

- Stack
  - A stack is used for holding the strings for the titles of each phase like flop and showdown. It was actually inconvenient to use this over an array of strings, vector, list, or any other non sorted forward container since you need to put in the last titles first due to it fi-lo nature. I just used it to include it on here for the assignment.
- Pair
  - The pair is the basis of the card class. My pair takes an enumerated value for the rank and suit of the card and stores them together.
- List
  - I used the doubly linked list to hold my players that are still at the game table. Since there is no STL circularly linked list I implemented two helper functions to use the list as a circularly linked list. Those functions returned the next and previous players in the list who did not have their player status marked as out. The link list, used circularly, was the obvious choice for storing the games players as the game is played with turns moving clockwise in a circle and the dealer and blinds moving the opposite way. It made it easy to loop through player by player as there was no need to randomly jump and skip across players.
- Vector

- Used for the deck of cards. This just holds the cards in a container, has them be shuffled with random_shuffle, and dispenses the card from the back of the deck as. This mimics the act of taking a deck of cards, shuffling them, and taking off the top of the deck as it would happen in an actual card game.

## Graphs

The players in the game are implemented as a graph. The graph made of players is a directed and unweighted unweighted graph. Each node in the graph is directed at the next node in the list until the last node circles around again to the node that is started at. The graph does not have any particular root but it always starts off on the node one clockwise of the human player and it can have anywhere between 2 and 6 nodes and since the graph is circular it always has an edge amount equal to the amount of nodes/ players.

## Trees

The hand of cards that the player holds is implemented using a red black tree. It is used because the player will only have a single of each card and it is better for the cards to be output in the card sorting order. Since the red black tree stores its nodes in order the output order of the cards is

sorted by suit and rank which makes viewing card hands easier to see and evaluate.

## STL Algorithms

- min() function used for bounds checking so a user could not spend more chips than they have
- random_shuffle() is used to shuffle the deck of cards after it is created so they are dealt in a random order.
- Sort() function used with iterators to sort various containters.
- make_pair() used to create the cards as they are implemented as pairs
- Iterators for looping with data structures and const_iterator for when I did not need to modify any container elements. I used the iterator returned from the erase() function when deleting from the list while looping, like removing players who lost, to ensure that I always had a valid iterator to the next element.

## Recursion / Recursive Sort

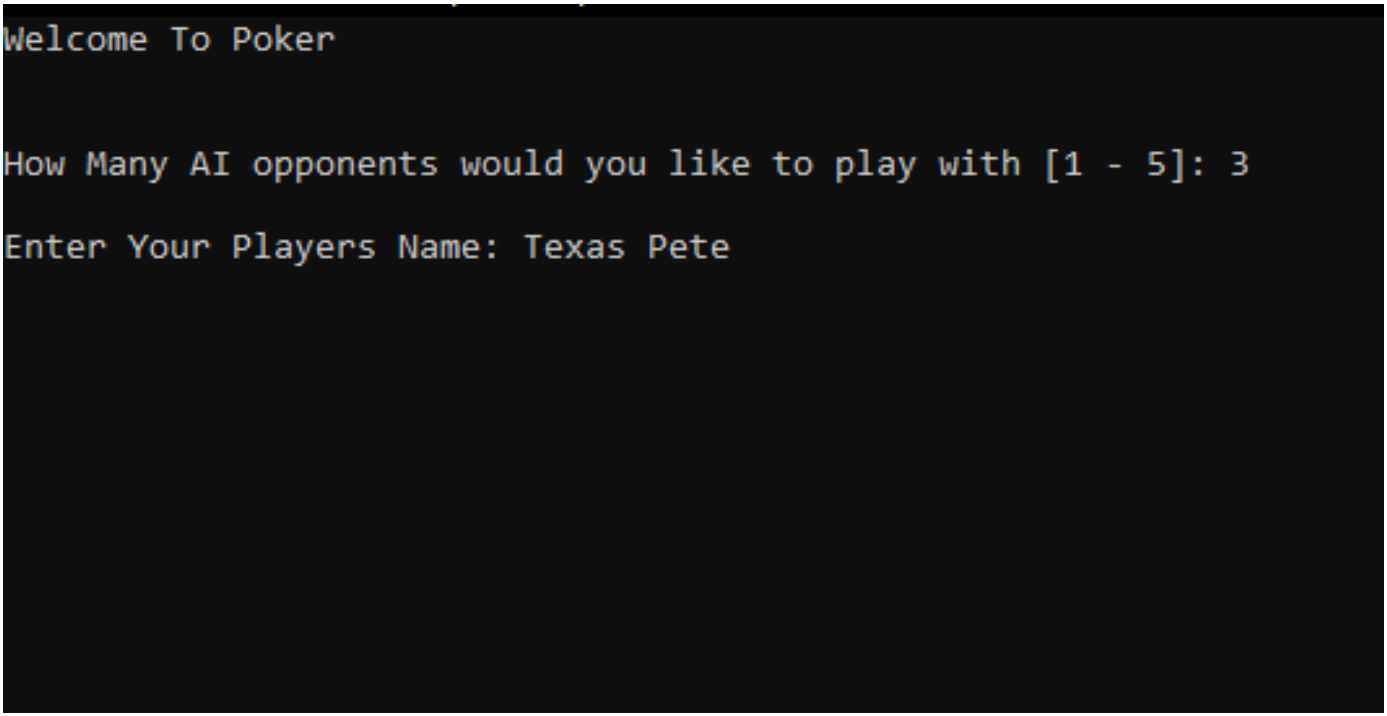
The game has a random name generating system for the AI players that assigns them random first and last name. When creating the names the arrays that hold possible first and last names are sorted recursively using a recursive implementation of the bubble sort. The recursive sort takes an array and size and returns under the base case of the size being one. If it

passes the base case then it linearly runs through the array swapping adjacent mismatched pairs and after it is done it calls itself again with the same array and the size minus one.

## Hashing

Hashing, as well as the recursion, is used to in the random name generating system. For hashing I used the std unordered_map for the hash table. This hash table held both the first and last name of the player to be returned from the function and a random name is accessed by giving a random index to the table and getting back first and last names. After the names gotten from the table they are returned from the function from the hash table to the player.

## Images Of Game Play



```
Enter Your Players Name: Texas Pete


-----------------------------------Blinds-----------------------------------
Big Blind: AI 2, 10 chips
Small Blind: AI 3, 5 chips


----------------------------------Pre Flop----------------------------------
Community Cards:
AI 1, Chips: 90, Move: Call 10
AI 2, Big Blind, Chips: 90, Move: Check
AI 3, Small Blind, Chips: 90, Move: Call 5

Your Turn. Your Chips: 100      Pot: 30 chips
Your Cards: [ Queen of Diamonds ] [ King of Spades ]
Choices:
1.) Call 10
2.) Raise 10
3.) Raise 20
4.) All In
5.) Fold
Your Choice: 1



Texas Pete, Dealer, Chips: 90, Move: Call 10


----------------------------------The Flop----------------------------------
Community Cards: [ King of Hearts ] [ 3 of Diamonds ] [ Jack of Hearts ]
AI 1, Chips: 90, Move: Check
AI 2, Big Blind, Chips: 90, Move: Check
AI 3, Small Blind, Chips: 90, Move: Check

Your Turn. Your Chips: 90       Pot: 40 chips
Your Cards: [ Queen of Diamonds ] [ King of Spades ]
Choices:
1.) Check
2.) Raise 10
3.) Raise 20
4.) All In
5.) Fold
Your Choice:
```

```
---------------------------------The Showdown---------------------------------

Not determining winner until final project. Chips just go to the player for now

Round Winner Is: Texas Pete
Winning Hand: [ Queen of Diamonds ] [ King of Spades ]  [ King of Hearts ] [ 3 of Diamonds ] [ Jack of Hearts ]
Enter anything to continue onto the next hand or q to quit: q


Results
Texas Pete.....Chips: 250
AI 2.....Chips: 50
AI 1.....Chips: 50
AI 3.....Chips: 50
Press <RETURN> to close this window...
```