Royal Flush

Software Specification

Authors: Kevin Lee, Ryan Kim, Li-Yu Ho, Joseph Balardeta, Daniel Jong, Aidan Woods

Software Version: 0.2 (BETA VERSION)

Affiliation: Royal Flush Team

Table of Contents

Software Specification 1

Table of Contents 2

Glossary 3

Glossary of Poker Terms 3
- General Game Terms 3
- Hand Ranking 4

Poker Server Software Architecture Overview 5
- Main data types and structures 5
- Major software components 5
- Module Interfaces 6
- Overall Program Control Flow 7
- Control Flow Chart 7

Player Software Architecture Overview 8
- Main data types and structures 8
- Major software components 8
- Module Interfaces 10
- Overall Program Control Flow 10

Installation 14
- System Requirements 14
- Setup and Installation 14
- Building, compilation, and installation 14
- Uninstalling 14

Documentation of packages, modules, interfaces 15
- Detailed description of data structures 15
- Detailed description of functions and parameters 18
- Detailed description of input and output formats 20
- Detailed description of the communication protocol 20

Development Plan and Timeline 21
- Partitioning of tasks 21
- Team member responsibilities 21

License and Disclaimers 22

References 23

Index 24

Glossary

“.c” - Describing a C file that contains a certain amount of functions pertaining to the program
“Enum” - Data type where all values we know are preset, but still part of the same variable.
“Linux” - The platform this project was programmed on, a basic operating system.
“Modules” - Modules contain small parts of the function that are slowly built out into the bigger picture of the program.
“Program” - Referring to the Chess program itself.
“Software” - Describes the set of .c files that make up the program.
“Struct” - Describes how certain elements of the game are defined.
“tasks” - Describes different elements of the program, like the main interface, the AI, etc. (also in reference to the partitioning for the group).

Glossary of Poker Terms

General Game Terms

Call - To contribute the minimum amount of points to the pot necessary to continue playing a hand.
Raise - To wager more than the minimum required to call, forcing other players to put in more points as well.
Fold - To give up by placing your cards face down on the table, losing whatever you have bet so far. You only fold when you think your hand is too weak to compete against the other players.
Check - To do nothing and wait for other player’s decisions. Checking can only be done when no player before has bet in this round.
Showdown - When, after the final round of betting, players turn their hands face-up. A poker hand will only reach a showdown if there are callers in the last round of betting, or if someone is all-in prior to the last betting round. The aim of the game is to make the best hand at showdown.
Hand - Five cards, made of a player's pocket cards and the community cards.
Pool - Where all the points being bet in the game will be placed.

Hand Ranking

1. ***Royal Flush*** - The best possible hand in Texas hold'em is the combination of ten, jack, queen, king, ace, all of the same suit.
2. ***Straight Flush*** - Five cards of the same suit in sequential order.
3. ***Four of a Kind*** - Any four numerically matching cards.
4. ***Full House*** - Combination of three of a kind and a pair in the same hand.
5. ***Flush*** - Five cards of the same suit, in any order.
6. ***Straight*** - Five cards of any suit, in sequential order.
7. ***Three of a Kind*** - Any three numerically matching cards.
8. ***Two Pair*** - Two different pairs in the same hand.
9. ***Pair*** - Any two numerically matching cards.
10. ***High Card*** - The highest ranked card in your hand with an ace being the highest and two beige the lowest.

Poker Server Software Architecture Overview

Main data types and structures

struct sockaddr_in - Server address needed to connect with.

struct hostent - Contains the server host information.

struct Connection - Contains connection information to the client (socket, port number).

Struct ServerConnection - Contains connection information to the server (socket, port number).

Major software components

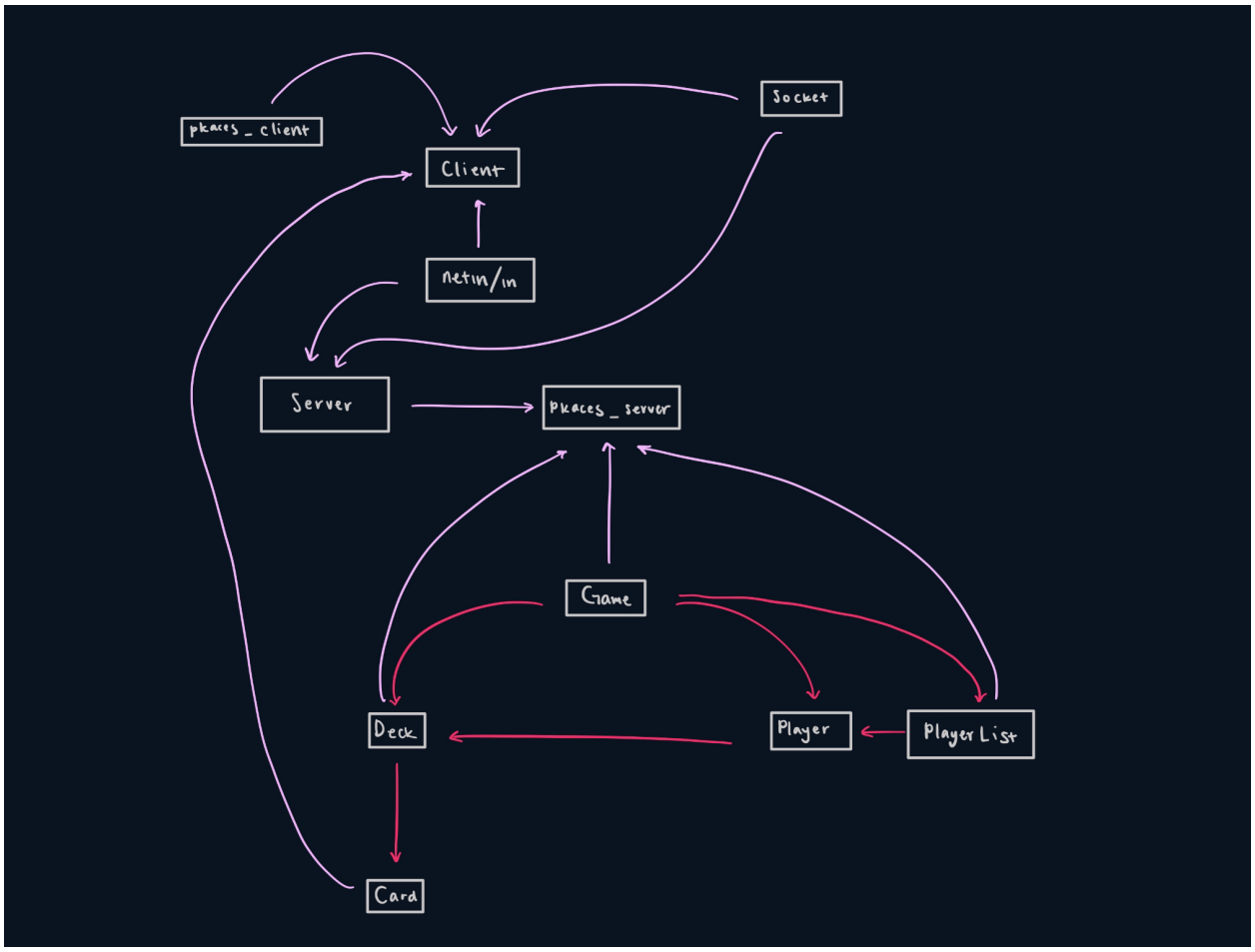
Server.c - Contains function for reading and writing from sockets occurs from the server side (or from the client).

pkaces_server.c - Where the main game “loops.”

pkaces_client.c - Establishes connection with the client to the server.

Client.c - Contains functions for reading and writing from sockets occurs from the client side (or from the server).

Basic Module Hierarchy



Module Interfaces

sys/types.h - Contains definitions of a number of data types used in system calls.

sys/socket.h - Includes a number of definitions of structures needed for sockets.

netinet/in.h - Contains constants and structures needed for internet domain addresses.

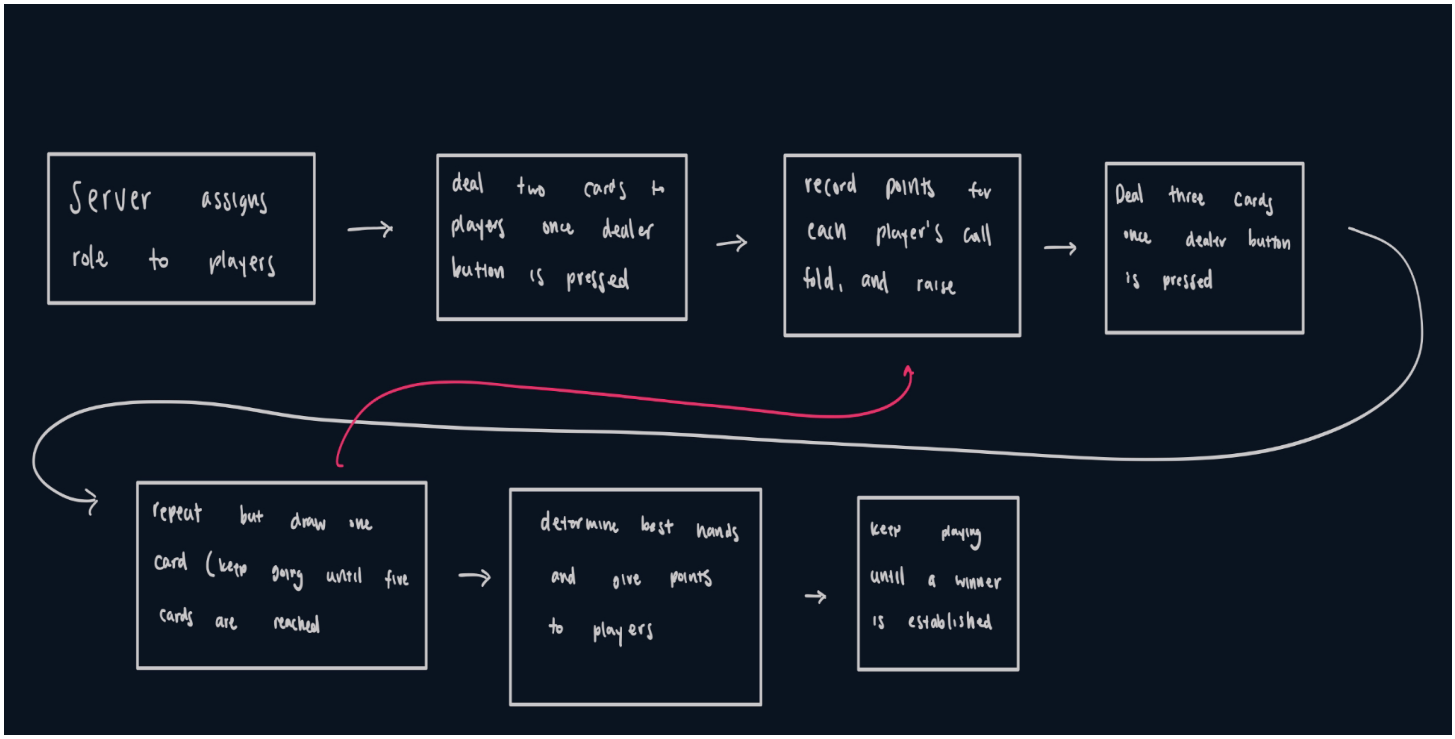
Server.h - Header file for Server.c. Defines the Server data structure.

Client.h - Header file for Server.c. Defines the Server data structure.

Overall Program Control Flow

1) The server assigns roles (big blind, small blind, dealer) to the players.
2) Wait for the dealer to click on the button to deal two cards to every player. Each player can only see their own cards.
3) Wait for the players to make their decisions until all players have folded or called while recording the points for each player and the amount of points in the pot. If any player runs out of points, the player loses and becomes watch-only.
4) Wait for the dealer to click on the button to deal three cards on the table which is visible to all players.
5) Repeat 3 and 4 except that it draws one card instead of three cards until there are five cards on the table. Keep checking whether there is only one player left in the game and if so, the player wins all the points in the pot.
6) If there are five cards on the table and there are more than two players in the game after the final bet, show the cards of the remaining players. Find the best hands for each player and compare them to see who is the winner. The winner takes all the points in the pot.
7) Repeat 1 through 6 until all players but one is eliminated.

Control Flow Chart



Player Software Architecture Overview

Main data types and structures

struct Game - The main storage unit of the game, contains the deck of cards players will play from, and the players that will be playing the game as well as the amount each player wishes to bet.

struct Player - The main storage unit for a player in memory. This data structure will include the number of points a player has, his or her name, and boolean integer variables that keeps track who is the Big Blind, the Small Blind, and the dealer (the “type” of player).

struct Deck - A doubly linked list of cards that keeps track of which card is at the top of the deck and at the bottom of the deck. Will link 52 cards by having each card point towards the card that comes before or after its placement in the deck.

struct DeckEntry - Will the contain the order of a given deck during a game.

enum Suit - Makes the suit of a card easier to read in code by assigning numbers from 0 to 3 to each suit.

enum Rank - Makes the number of a card easier to read in code by assigning numbers from 1 to 13. 1 will be assigned to Ace, and the face cards (Jack, Queen, King) will be assigned 11 to 13 consequently.

struct Card - the main storage unit of each poker card in the deck (contains its suit and rank).

Struct PlayerList - Keeps track of which player does what (such as a player or the dealer).

Struct PlayerEntry - Contains the order of the participating players in a given instance of a game.

Major software components

Card.c - Contains functions for creating, deleting, and cloning the cards.

Deck.c - Contains functions for initializing a doubly linked list of 52 cards, shuffling the deck, picking a card from the deck, deleting the deck.

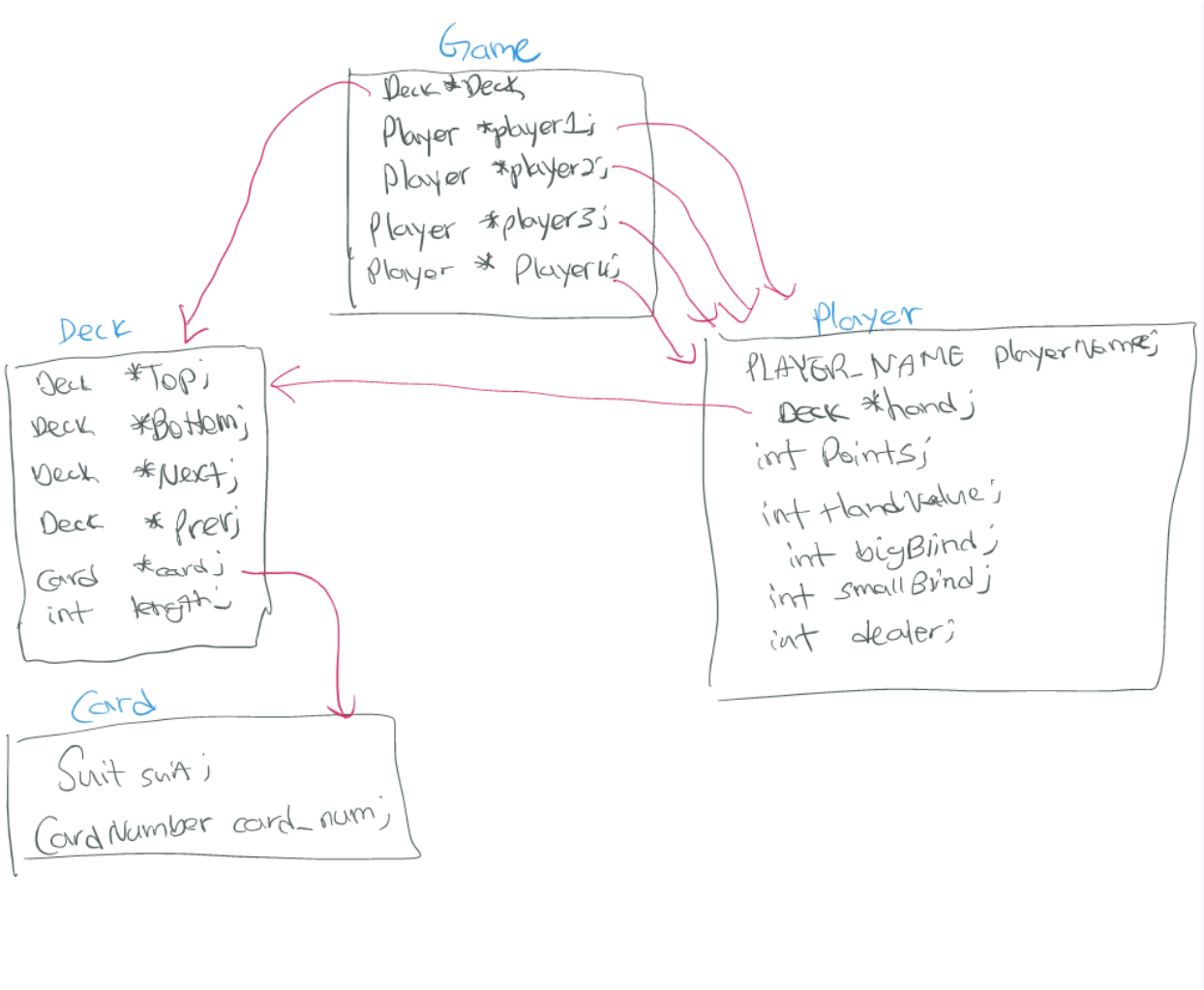
Game.c - Contains functions for creating and deleting the game and handles the main “looping” of the entire game.

Player.c - Contains functions for creating and deleting a player.

GUI.c - Contains the functions for the GUI graphics/options.

PlayerList.c - Contains the functions to deal with the players and the dealer (creating and removing players).

Basic Module Hierarchy



Module Interfaces

Card.h - Header file for Card.c. Defines the Card data structure.

Deck.h - Header file for Deck.c. Defines the Deck data structure.

Game.h - Header file for Game.c. Defines the Game data structure.

Player.h - Header file for Player.c. Defines the Player data structure.

Client.h - Header file for Client.c. Defines the Client data structure.

GUI.h - Header file for GUI.c. Defines the GUI data structure.

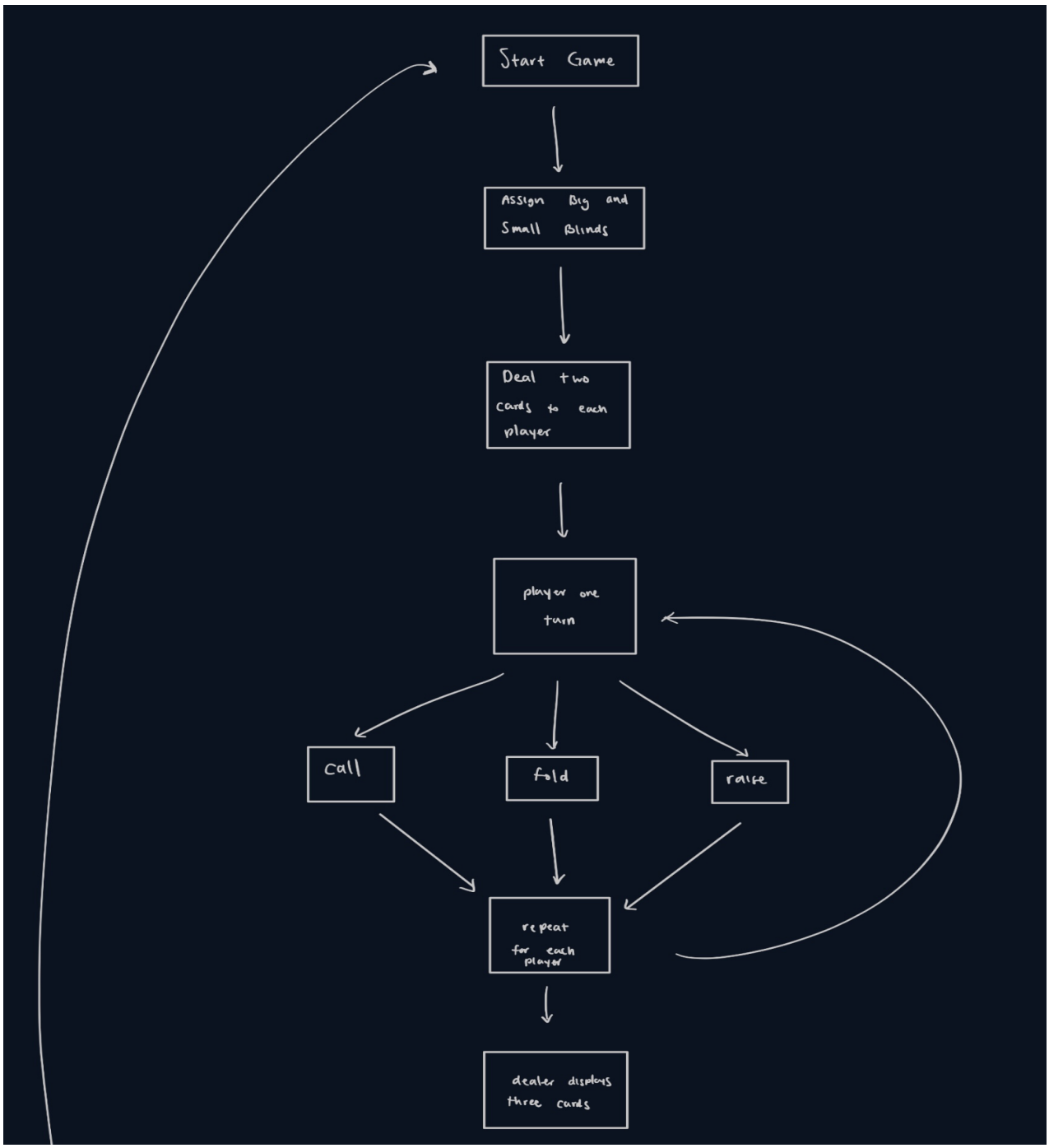
PlayerList.h - Header file for PlayerList.c. Defines the PlayerList data structure.

Overall Program Control Flow

1) The first player will be assigned as big blind and the last player will be assigned as small blind. The big blind will be required to bet 2 points while the small blind will be required to bet half of the points the big blind put in.
2) Two cards will be dealt to every player and only the user's cards will be shown face up.
3) Players will be asked to either call, raise or fold consequently, and the player being the big blind will be asked last.
    a) If the player decides to call, the player will bet the number of points the previous player will bet.
    b) If the player decides to raise, the player will be asked to bet the number of points that is higher than the previous highest number of points.
    c) If the player decides to fold, the player will be skipped for the remainder of the game.
4) The dealer will then take out three cards from the deck and display them on the table.
5) Players will be asked to either check, call, raise or fold consequently until every player has called with the highest amount of points or folded, and the player being the big blind will be asked first.
    a) Checking can only be done if no one before the player has bet. If the player decides to check, the player does nothing.
    b) If the player decides to call, the player will bet the number of points the previous player has bet.
    c) If the player decides to raise, the player will be asked to bet the number of points that is higher than the previous highest number of points.
    d) If the player decides to fold, the player will be skipped for the remainder of the game.
6) The dealer will then take out a card from the deck and display it on the table.

7) Repeat step #5 and #6 until there are five cards on the table or there is only one player remaining before the end of the game.
8) At the end of the game, if there are more than one player in the game, the game will show the hands of every player and decide who the winner is. The player who has the highest ranking of hands wins. If not, the only player automatically wins the game.
9) The winner will then collect the points in the pool.
10) The menu will ask players if they want to play the game again. The player with zero or less points will not be allowed to play again.

Control Flow Chart




player one turn
call
fold
raise
repeat for each player
dealer displays card
repeat process and eliminate players
determine winner and give points to player
player decides on playing again

Installation

System Requirements

- Linux OS running gcc version 1.7.1 or newer
- 100mb of disk space
- 100kb of free RAM
- Stable internet connection

Setup and Installation

There is no setup required besides the instructions in the **Building, compilation, and installation** section.

Building, compilation, and installation

To set up the Royal Flush program, type ‘tar -xvzf Poker_Beta_src.tar.gz’, then ‘cd Poker_Beta_src’. Launch the ‘pkaces_client’ and the ‘pkaces_server’ executable files in order to launch the game. If any issues occur or for more details, please refer to the user manual.

Uninstalling

To uninstall the program, launch the ‘pkaces_client’ and the ‘pkaces_server’ executable files. Then in the menu, select the option that says “Uninstall” in order to uninstall and exit the program.

Documentation of packages, modules, interfaces

Detailed description of data structures

```
/* Data structure of Game that contains a deck of cards and players playing the game. */
typedef struct {
	PLIST *players;
	Player *Dealer;
	DECK *boardCards;
	int betPoints;
} Game;

/* Data structure of Player that contains type of player, state the player is in, deck, and points */
typedef struct {
	int id;
	TYPE type;
	P_STATE p_state;
	DECK *deck;
	int points;
	Connection *connection;
} Player;

/* Data structure of a linked list of Deck */
typedef struct {
	Deck *Top;
	Deck *Bottom;
	Deck *Next;
	Deck *Prev;
	Card *card;
	int length;
} Deck;

/* State of player (play or fold) */
typedef enum {PLAYING, FOLDED} P_STATE;

/* Type of player (dealer or player) */
typedef enum {PLAYER, DEALER} TYPE;

/* 0 represents clubs, 1 represents hearts, 2 represents spades, 3 represents diamonds */
typedef enum {CLUBS = 0, HEARTS, SPADES, DIAMONDS} Suit;
```

```
/* 1 represents ace, and the enum goes till 13 for each rank */
typedef enum {ACE = 1, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK,
QUEEN, KING} Rank;


/* Contains the suit and rank of a specified card. */
typedef struct {
        Suit suit;
        Rank rank;
} Card;

/* A structure containing an internet address. This structure is defined in netinet/in.h. */
typedef struct
{
        short sin_family;               /* must be AF_INET */
        u_short sin_port;
        struct in_addr sin_addr;
        char sin_zero[8];               /* not used, must be a zero */
} sockaddr_in;

/* Contains the host information */
typedef struct {
        char *h_name;           /* official name of host */
        char **h_aliases;       /* alias list */
        int h_addrtype;         /* host address type */
        int h_length;           /* length of address
        char **h_addr_list;     /* list of addresses from name server */
        #define h_addr H_addr_list[0]           /* address, for backward compatibility */
} hostent;

/* A structure containing socket information, port number, and address (server side) */
typedef struct {
      int sockfd, portno;
      char buffer [256];
      Sockaddr_in serv_addr;
      Host *server;
} ServerConnection;
```

```
/* A structure containing player information from the client side */
typedef struct {
	int points;
	PLAYERSTATE p_state;
	Card *card1;
	Card *card2;
	ServerConnection *connection;
} ClientPlayer;

/* A structure containing the game's information from the client side */
typedef struct {
	char playerData[256];
	int betPoints;
	ClientPlayer *user;
	int gameOver;
} ClientGame;

/* A structure containing the deck in a given instance of a game */
struct Deck {
	DENTRY *First;
	DENTRY *Last;
	int Length;
} DECK;

/* A structure containing the order of the deck in a given instance of a game */
struct DeckEntry {
	Card *Card;
	DECK *Deck;
	DENTRY *Next;
	DENTRY *Prev;
} DENTRY;

/* A structure containing the players in a given instance of a game */
struct PlayerList{
	PENTRY *First;
	PENTRY *Last;
	int Length;
} PLIST;
```

```
/* A structure containing the order of the players in a given instance of a game */
struct PlayerEntry {
	PLIST *pList;
	PENTRY *Next;
	PENTRY *Prev;
	Player *Player;
} PENTRY;

/* A structure containing socket information, port number, and address (client side) */
typedef struct {
	Sockaddr_in serv_addr;
	Sockaddr_in cli_addr;
	socklen_t clilen;
	int sockfd, newsockfd, portno;
	char buffer[256];
} Connection;
```

Detailed description of functions and parameters

```
/* continuously running loop that plays the game until the game meets its requirement to terminate */
int main();

/* prints the menu that the user will get to interact with by choosing from different from */
void PrintMenu();

/* prints the current working poker table that displays the hand of each player *.
void PrintTable();

/* allocates a doubly linked list of a deck of cards and four players */
Game *CreateGame();

/* Frees the memory space for the deck and players */
void DeleteGame(Game *game);

/* Deletes the doubly linked list *hand if a player decides to fold during the game */
void Fold(Game *game);

/* checks for the winning hand at the end of the game */
void CheckWinner(Game *game);

/* Determines the winner and cleans up memory */
void EndGame(Game *game);
```

```
/* Responsible for a single instance of a game */
void GameLoop(int option);

/* Creates a player with his or her desired player name */
Player *CreatePlayer(PLAYER_NAME playerName);

/* returns 1 if the player has enough points to either call or raise, returns 0 otherwise */
int pointCheck(Player *player);

/* Checks what kind of ranks each player's hand has */
void CheckRank(Player *player);

/* Deletes player at the end of the game */
void DeletePlayer(Player *player);

/* Randomly chooses which player gets assigned as the big blind */
void chooseBigBlind();

/* Randomly chooses which player gets assigned as the small blind */
void chooseSmallBlind();

/* Randomly chooses which player gets assigned as the dealer */
void chooseDealer();

/* Creates a doubly linked list of cards into a deck of cards */
Deck *CreateDeckEntry(Card *card);

/* Appends a card at the end of the deck /*
void AppendCard(Deck *deck, Card *card);

/* Randomizes the order of the doubly linked list of cards */
Deck *ShuffleDeck(Deck *deck);

/* Updates the deck once a card has been drawn from the deck */
Deck *UpdateDeck(Deck *deck);

/* Deletes the doubly linked list of cards and frees the memory space of the deck */
void DeleteDeck(Deck *deck);

/* Creates a card with the suit and the card number passed in the parameter */
Card *CreateCard(Suit suit, CardNumber card_num);

/* Deletes the card */
void DeleteCard(Card *card);
```

```
/* called when a system call fails. Displays an error message and then aborts the program */
void error(char *msg);

/* used to access the record of a certain player on the server */
void accessRecord(Player *player);

/* used in Client.c to establish a client's connection to the server */
void clientConnection();

/* takes a client's port code and uses it to establish a connection with the server */
void serverConnection(char[] Portcode);
```

Detailed description of input and output formats

Within the GUI, the player's inputs will be determined by buttons they click on. There will be buttons, round depending, to check, raise, call, and fold. These decisions that the player/AI makes are not logged anywhere.

Detailed description of the communication protocol

In communication with the poker server, the flags will determine the state of connectivity with the server, and settings will determine aspects of the connection that the user can tailor to how they want to use the server. They describe states of connection mostly, flags could be telling the user their port code is being sent to the server, their connection is being finalized, their game is getting synchronized with the server, etc.

Within code, function calls will be made to establish a connection to the client. Functions communicate with each other to fetch specific info for connections. For example, the port code of the player will be sent in a function to the server as a method of establishing a connection for that particular player. Another example is getting the address of the server that we are connecting to.

Development Plan and Timeline

Partitioning of tasks

Software/Documentation Subteam (2 members of the team)

- Maintain Software Specification and User Manual, keep them up to date with the current code.
- Program some of the more minor aspects of the project.

Software Subteam (3 members of the team)

- Program the more advanced parts of the project.

Software Framework Manager (1 member of the team)

- In charge of organizing the code structure for the team and overseeing all code operations to keep things running smoothly.

All members:

- Testing of code before releases (debugging, valgrind).
- Feedback (suggestions).

These tasks are subject to slight changes based on the workload that each subteam has.

Team member responsibilities

- All members are responsible for testing/compiling any new changes made before software submissions.
- All members should inform others of said changes.
- All members should attend/participate in scheduled meetings regarding development.

License and Disclaimers

Presented images may not represent the actual program

References

All references are from resources/material published by Professor Doemer (specifically for the EECS 22 and EECS 22L courses of WINTER 2022 and SPRING 2022)

Index

.c: 4, 6, 8, 9, 10, 17
enum: 4, 7, 8, 14
Linux: 4, 12
modules: 4, 13
Program: 4, 6, 9, 13, 17, 18, 19
software: 4, 6, 7, 8, 18, 19
struct: 3, 5, 7, 8, 13, 15
tasks: 4, 18, 19