Project 2

Game: Go Fish: Extended

Course: CSC-17C-C++
Section: 43673
Due Date: June 6 2024
Name: Yasmeen Allahaleh

1 Introduction:

1.1 What are you coding and why did you choose this Game?

I decided to code the card game Go Fish. I decided that I wanted to choose this game because I used to play this game with my younger siblings whenever we were little, so I was familiar with it.

1.2 How long did you spend, how many lines, classes, etc.?

I've roughly spent about a week and a half coding this game. This game contains about roughly 868 lines of code.

1.3 Where on github is it located?

It is located on my Github : **https://github.com/Yallahaleh** in the CSC_17C Repository. The project is in a folder called Project_17C_Lehr and the final coded version of the project is named "Allahaleh_Yasmeen_Project2_17c".

2 Approach to Development:

2.1 Concepts

Concepts used:

a.) Containers:

a.1.) stack: I used stacks to store the history each outcome of the round such as "Won" or "Lost"

a.2.) list: I used list to hold each player's card set that used to play

a.3.) queue: I used a queue to store the counts of loses and wins of the player

a.4.) map: I used a map to store the player's inserted name and the amount of coins they have

a.5) set: I used sets to store each set of pairs the player's had in their deck

b.) Algorithms

- I utilized algorithms in the development of my game such as find to find certain cards, swap and sort in order to deal with the cards and the decks more efficiently.

c.) Tree/Graphs

(Can be found in Game.cpp file on lines 27-87 & 206-212)

- I utilized this concept in order to display which cards the player can receive. Each card can be displayed from the tree and are connected to each other.

d.) Recursion

(Can be found in Game.cpp on lines 231 and 235)

- Recursion was used in order to give each player their set of cards.

e.)Recursion Sort

(Can be found in Game.cpp on line 237)

- I used a bubble recursion sort in order to sort out the player's card to make it easier for them to navigate which and how many cards they have.

3 Game Rules:

3.1 Game Rules

The rules of the game are quite simple and usually consist of 2 to 5 players. To begin, a dealer passes 5 cards to each individual player from the shuffled deck. Once everyone receives 5 cards, the leftover cards from the deck then get placed in the middle. The player to the left of the dealer goes first. The goal is to get as many sets of 4 cards with the same number. So the first player then asks any other player for the value they want. If the player they ask does not have it, that player calls Go Fish and the player who asked draws a card from the deck and the turn goes on to the next player. But if the player they asked does have that value, they must give them all the cards they have with that value. If this happens the player who asked gets to go again. This is repeated until the main deck runs out of cards or any other player runs out of cards. When the game is finished, whoever has the most sets of the same cards wins the game.

4 Description of Code:

4.1 Organization:

My code has been organized into three different .cpp files.

One of the files is called "Card functions". This .cpp file is where I store all of the functions that relate to passing, creating, shuffling, and checking if the deck is empty.

My second .cpp file that is named "GoFish" contains all the necessary functions that relate to playing the game itself such as asking for cards, the AI players asking for cards, pulling from the deck, saying "Go Fish", and checking if you have a set of numbers during each turn.

My last .cpp "Game" is where the game menu is, where you choose a 2 or 3 player mode, see winning history, and actually play the game. These options are organized into switches where each separate choice lies in. This is also the file where you keep track of your player name and coins. Once the player runs out of coins, the player can no longer play and the program says goodbye. This function also holds the classes and functions for the tree that displays the card the user can receive.

5 Sample Input/Output:

```
--------------------------------------------------

These are the cards you can receive:
1 2 3 4 5 6 7 8 9 10

========== GoFish! =============


Here are your cards yaz: 1 10 10 10 10


You found a set of 10 in your Deck!


     ******** Your Turn *********
-----------------------------------------
Here are your Cards:
1
Asking AI for a value . . .
What value do you want to ask Player AI? :
```

** Here is and output of the tree displaying what cards the user can receive*

```
------------------------------------------------------

These are the cards you can receive:
1 2 3 4 5 6 7 8 9 10

========== GoFish! =============


Here are your cards yaz: 4 9 9 10 10


     ******** Your Turn *********
-----------------------------------------------
Here are your Cards:
4 9 9 10 10
Asking AI for a value . . .
What value do you want to ask Player AI? :
9
AI is looking through His Cards . . .
AI found the card you're looking for : 'Here you go . . .'
He had : 2 card(s) with the value of: 9


You found a set of 9 in your Deck!

     ******** Your Turn *********
-----------------------------------------------
Here are your Cards:
4 10 10
Asking AI for a value . . .
What value do you want to ask Player AI? :
```

*Here is an example of the cards being used in the recursion sort and the cards being given by recursion.

6 FlowChart:

START

cout: "Welcome, this game costs 20 coins, do you want to play?"

If('y')

If(!'y)

cout: "Welp.. goodbye then!"

Displays menu option
"1.) Play Go fish with 1 bot
2.) Play Go fish with 2 bots
3.) See winning/Lose History "

if(choice != 1, 2 or 3)

cout: "Invalid Input"

If(choice = 1)

if(choice =3)

cout "----Wins/Lost History---"
cout " Wins:" << wltracker.front()
cout << "Loses:" << wltracker.back()

While(!winsLose.empty())

cout << winsLose.top() << endl;
winsLose.pop();

coins = coins - 20

create 3 list<int>
player deck, AI deck,
and main deck

tree.card(tree.root, cards)
cards++
tree.display(tree.root)

createDeck()
shuffle(Deck)

m = 0
n = 0
rec(m,n,playerDeck, deck)
Rec(m,n,AIDeck,deck)

recSort(playerDeck, 5)

cout "Here are your cards"
displayDeck(playerDeck)

checkPair(playerDeck, playerpair)
checkPair(AIDeck, AIpair)

while(p != 1)

False

setYou = playerPairs.size()
setAI = AIpairs.size()

if(setYou > setAI)

winsLose.push("Won");
wins++;
wltracker.pop();
wltracker.pop();
wltracker.push(wins);
wltracker.push(loses);

cout << "You Win This Round!"

else if(setYou < setAI)

winsLose.push("Lose");
wins++;
wltracker.pop();
wltracker.pop();
wltracker.push(wins);
wltracker.push(loses);

cout << "You Lost"

if(setYou > setAI)

winsLose.push("Won");

cout << "Its a draw!"

askAI(playerDeck, AIDeck, deck, playerPairs, AIpairs)

for loop (i < 5)
{ checkPair(playerDeck, playerpair)
checkPair(AIDeck, AIpair)}

p = checkEmpty(playerDeck, AiDeck, 1, deck)

cout "Here are your cards" <<name;
display(playerDeck)

True

aiAsks(playerDeck, AIDeck, deck, playerPairs, AIpairs)

for loop (i < 5)
{ checkPair(playerDeck, playerpair)
checkPair(AIDeck, AIpair)}

p = checkEmpty(playerDeck, AiDeck, 1, deck)

cout "Here are your cards" <<name;
display(playerDeck)

if(option = 'y')

cout << "Do you want to play again"

cin >> option

if(option != 'y')

if(option != 'y')

cout << "Do you want to play again"

cin >> option

if(option= 'y')

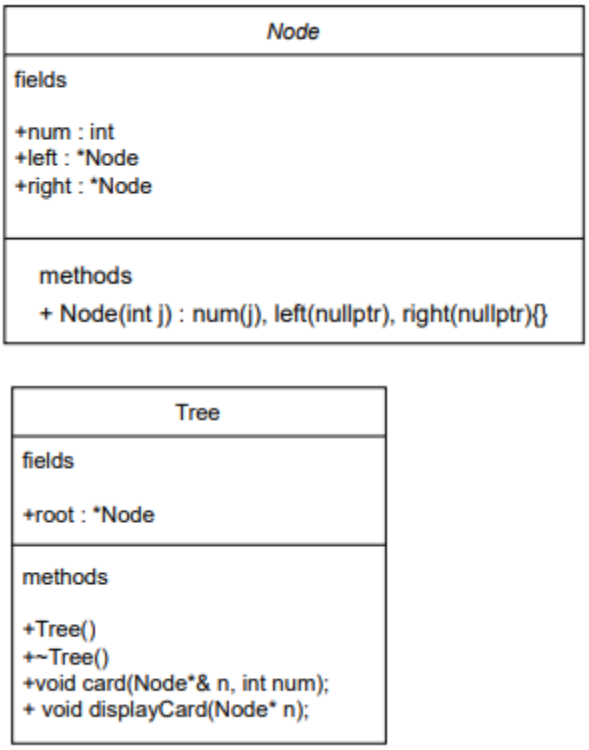
if(menu = 'y')

cout << "Do you want to go to the menu?

if(menu != 'y')

cout << "GoodBye !"

7 UML Class Diagram:



8 Documentation of Code:

***.1 Pseudo Code:**

//PseudoCode:
"Game.cpp"

//create a class for Node
//make public:
//integer number
//Node* for the left
//Node* for the right
//Node(int j) : num(j), left(nullptr), right(nullptr){}

//create class for tree
//make public:
//Create Node* root
//Tree()

```
//~Tree()
//  create void card(Node*& n, int num);
 // create void displayCard(Node* n);

//Tree::Tree() : root(nullptr){}

//define void destroy(Node* n){
//if n does not equal nullptr
//destroy(n->left);
//destroy(n->right);
//delete n;

//Tree::~Tree(){
// destroy(root);
//root = nullptr;



//void Tree::card(Node*& n, int num)
// If n equals nullptr
 //n equals new Node(num)
//else if num is less than n->num
// card(n->left, num)
//else
//card(n->right, num)

//define void Tree::displayCard(Node* n)
 //If n does not equal nullptr
//displayCard(n->left)
//cout  n->num
//displayCard(n->right)


//Set set random number seed
//int coins =100;
//string name;
//stringanswer;

//ask player to insert name
// cin the name
```

```
//create queue<int> called wllTracker to track num of wins and losses
//create stack<string> called winsLose to track outcome of each round

//int wins = 0;
//int loses =0;

//push num of wins into the queue<int> wlTracker.
//push num of loses into the queue<int> wlTracker.

//create map <string, int> called mp to store player name and coins
//insert map[name] = coins

//start of do while loop 1
{

	//if coins == 0, then break out of the do-while loop

	//cout player name and ask if they want to play and the game costs 20 coins
	//cin  answer

	//if answer is not equal to 'y', then cout "Bye"
	//return 0;

	//display the Map that contains player name and current coins
	//Display menu options
	// cout - "1.) Play GoFish! Against 1 bot"
	// cout - "3.)See History"

	//int choice;
	//cin your choice

	//switch(choice)
		//case 1:
		{
			// string option

			//start of do-while loop 2
			{
				//subtract 20 coins from players total coins;
```

```
//display the map that contains player name and new amount
of coin

//cout “These are the cards you can receive”

//start for loop and iterate 10 times
//tree.card(tree.root, cards)
//increment cards
//end of for loop

//cout  “Game in Session”
//cout “========== GoFish! ==========”

//create list<int> (for main Deck) = new list<int>
//create list<int> (for player Deck) = new list<int>
//create list<int> (for AI Deck) = new list<int>

//create set<int> (to store sets of same value for player) = new set<int>
//create set<int> (to store sets of same value for AI player) = new set<int>

//createDeck(list of main deck)
//shuffle(list of main deck)

//int m equals 0
// int n equals 0
//rec(m , n, playerDeck, 5)

//m equals 0
// n equals 0
//rec(m , n, AIDeck, 5)

//recSort(playerDeck, 5)

//cout “here are your cards
//display the player’s cards dealt

//check for any matching sets in player deck
//check for any matching sets in AI’s deck

//int p =0;
```

```
//start of do while loop 3
{
   //function
   //askAI for value(playerDeck, AIDeck, deck, playerpairs, Aipairs)

   //start of for loop that loops 5 times
  //(function) check player deck for any matching sets
  //(function) check AIdeck for any matching sets

  //p = check of any of the main, player, or Ai decks are empty
  //if p is equal to 1 exit out of do-while loop 3

  //cout "Here are your cards"
  //Function
  //displayDeck(playerDeck)

    //function
   //Ai asks you for value(playerDeck, AIDeck, deck, playerpairs, Aipairs)

   //start of for loop that loops 5 times
  //(function) check player deck for any matching sets
  //(function) check AIdeck for any matching sets

  //cout "Here are your cards"
  //Function
  //displayDeck(playerDeck)

  //p = check of any of the main, player, or Ai decks are empty
}while p does not equal 1;

//int setYou = size of the player's set called playerpairs
//int setYou = size of the AI's set called Aipairs
//if(setYou > setAI)
// {
//     push "Won" into the stack called winsLose
//     increments wins
//     pop from wlTracker queue twice
//     push the wins variable into wlTracker
//     push  the loses variable into wlTracker
//     cout " You win this round"
// }
```

```
                    // else if (setYou < setAi)
                    // {
                    //     push “Lost” into the stack called winsLose
                    //     increments wins
                    //     pop from wlTracker queue twice
                    //     push the wins variable into wlTracker
                    //     push  the loses variable into wlTracker
                    //     cout “ You Lost to AI?”
                    // }
                    //else
                    //{
                    //     push “Draw” into the stack called winsLose
                    //     cout “Its a draw”
                    //}

                    //if (coins == 0)
                    //{
                    //    cout “No more coins, so you can’t play anymore!”
                    //    break out of switch case 1
                    // }

                    //cout “Do you want to play again?”
                    //cin option

                    //if(tolower(option[0] == ‘y’)
                    //cout “New Round Created”

            //}while(tolower (option[0] == ‘y’) //end of do while loop 2

            // break;
  //}//end of case 1
        //case 2:
//{
                    // cout << "=============== Win/Lost History =============="
                    // cout "Total Wins: " <<  wltracker.front()
                    //  cout “Total Loses: " << wltracker.back()

                    //  cout "---------- Track Each Round Outcome ---------"

                    //  start of while loop
```

```
                        //   while (winsLose stack is not empty)
                        //   cout << winsLose.top()
                        //   pop stack winsLose stack
                       // }end of while loop

                       //  cout "----------------------------------------"

                       break;
        //}
        // default:
        //                cout “Invalid input”

      }//end of switch

     //if(coins == 0)
     //  {
           // cout "You have exhausted your Coins . . ."
           // break;
     //  }

    // cout "Would you like to go back to the menu?"
    //  cin menu;

  // }while(tolower(menu[0]) == 'y');   //end of do while loop 1

   // if(coins == 0)
  //    {
          // cout  "GOOD-BYE!"
         // break;
  //     }

}// end of program

//Functions coded:

//Prototypes for functions from CardFunction.CPP
void createDeck(list<int> *deck); //creates the deck
void shuffle(list<int> *deck); // shuffles the deck
void displayDeck(list<int> * play); //displays the deck
int disperse(list<int> *cards); // give a player a card
```

```
int checkEmpty(list<int> *play, list<int> *opp, list<int> *opp2, int dis, list<int> *deck);
void gameRules(); // checks if the decks are empty
void displayCoins(map<string, int> a, string name, int coins); // display the map that display the player and coins

//Prototypes for functions from GOFISH.CPP
void addRemoveCards(list<int> *play, list<int> *opp, list<int> *deck, int value); //add and remove card from deck
void askAI(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b); // coded for the player (you) to ask for a card for 2 player mode
void aiAsks(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b, string name); //coded for AI to ask you for a card for 2 player mode
void askAIThree(list<int> *play, list<int> *opp,list<int> *opp2, list<int> *deck, set<int> *a, set<int> *b, set<int> *c); // coded for the player (you) to ask for a card for 3 player mode
void aiAsksThree(list<int> *play, list<int> *opp, list<int> *opp2, list<int> *deck, set<int> *a, set<int> *b, set<int> *c, int d, string name); //coded for AI to ask you for a card for 3 player mode
void goFish(list<int> *play, list<int> *deck); // player pulls a card when opposing player says goldfish
void checkPair(list<int> *play, set<int> *pair, int b); //checks if a deck has a matching set
```

"Game".cpp

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

#include <cstdlib>
#include <iostream>
#include <ctime>
#include <iomanip>
#include <map>
#include <list>
#include <set>
#include <stack>
#include <queue>
#include <algorithm>
#include <iterator>

using namespace std;
```

```
/*File:   Game.cpp
 *Author: Yasmeen Allahaleh
 */
//Prototypes for functions from CardFunction.CPP
void createDeck(list<int> *deck);
void shuffle(list<int> *deck);
void displayDeck(list<int> * play);
int disperse(list<int> *cards);
int checkEmpty(list<int> *play, list<int> *opp, list<int> *opp2, int dis, list<int> *deck);
void gameRules();
void displayCoins(map<string, int> a, string name, int coins);

//Prototypes for functions from GOFISH.CPP
void addRemoveCards(list<int> *play, list<int> *opp, list<int> *deck, int value);
void askAI(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b);
void aiAsks(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b, string name);
void askAIThree(list<int> *play, list<int> *opp,list<int> *opp2, list<int> *deck, set<int> *a, set<int> *b,
set<int> *c);
void aiAsksThree(list<int> *play, list<int> *opp, list<int> *opp2, list<int> *deck, set<int> *a, set<int>
*b, set<int> *c, int d, string name);
void goFish(list<int> *play, list<int> *deck);
void checkPair(list<int> *play, set<int> *pair, int b);


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

    //set random seed time
    srand(static_cast<unsigned int>(time(0)));

    //starting amount of coins for player
    int coins = 100;
    //for player to answer if they want to play
    string answer;

    // asks the player their name
    string name;
    cout << "Please enter your name player: ";
    cin >> name;

    //Queue that's stores number of wins in front, and losses in back
    queue<int> wltracker;
    //stack stores the outcome of each round, such as "Won" or "lost"
    stack<string> winsLose;

    //Tracks user's wins and losses
    int loses = 0;
    int wins = 0;

    //pushes values into queue
    wltracker.push(wins);
    wltracker.push(loses);
```

```
//for player to answer if they want to
//go back to menu
string menu;

//Map stores player name and amount of coins
map<string, int> mp;
mp[name] = coins;

//Start of program output
do
{
   cout << "Hi " << name << ", you need coins to play this game of Go Fish!" << endl;

   // if player has no coins
   //they cant play anymore
   if(coins == 0)
   {
      cout << "You ran out of coins! No more gaming . . . GOODBYE !" << endl;
      return 0;
   }


   cout << "-----------------------------------------------" << endl;
   cout << "This Game Costs : 20 coins . . ." << endl;
   cout << "Do you want to play?" << endl; // asks if player wants to play
   cin >> answer;

   if(tolower(answer[0]) != 'y') // if no then exit program
   {
      cout << endl << "Welp bye then . . ." << endl;
      return 0;
   }

   //Display the map with player name and amount of coins
   displayCoins(mp,name,coins);
   cout << endl << endl;

   cout << "======== Menu ========== " << endl;
   cout << "Please choose an option:" << endl;
   cout << "1.) Play GoFish! against 1 AI bot" << endl; // cin >> 1 for this option
   cout << "2.) Play GoFish! against 2 AI bots" << endl; // cin >> 2 for this option
   cout << "3.) See History" << endl; // cin >> 3 for this option
   cout << "========================" << endl;
   cout << "Choice: ";
   int choice;
   cin >> choice; // where player makes choice

   switch(choice)// for menu options
   {
```

```
        case 1: // choic to play against 1 bot
        {
           string option;

           do
           {
              coins = coins - 20; // subtract 20 to play game
              displayCoins(mp,name,coins);// displays user's name and amount of coins
              cout << endl;


              cout << "Great! Game in session . . .  " << endl << endl;
              gameRules();
for(int i =0; i < 10; i++)
              {
                 tree.card(tree.root, cards);
                 cards++;
              }

              tree.displayCard(tree.root);
              cout << endl << endl;

              cout << "========== GoFish! ============" << endl;

              list<int>* deck = new list<int>(); // creates main deck
              list<int>* playerDeck = new list<int>(); // creates player deck
              list<int>* AIDeck = new list<int>(); // creates Ai deck

              set<int>* playerpairs = new set<int>(); // set to hold the matching card for player(you)
              set<int>* Aipairs = new set<int>();// set to hold the matching cards for AI player


              //Start Game
              createDeck(deck); // creates the deck
              shuffle(deck); // shuffle deck

              int m = 0;
              int n = 5;
              rec(m, n, playerDeck,deck);

              m = 0;
              n = 5;
              rec(m, n, AIDeck,deck);

              recSort(playerDeck, 5); // sorts player deck to make it easier to track cards


              cout << endl << endl<< "Here are your cards " << name << ": ";
              displayDeck(playerDeck); // display your deck

              cout << endl;
```

```
checkPair(playerDeck, playerpairs, 0); // check if you have any matching cards
checkPair(AIDeck, Aipairs, 1); // check if AI has any matching cards
cout << endl;

int p = 0; //track value for empty decks

do
{
   askAI(playerDeck, AIDeck, deck, playerpairs, Aipairs); // players turn

   for(int i =0; i < 5; i++)
   {
      checkPair(playerDeck, playerpairs, 0);// check if you have any matching cards
      checkPair(AIDeck, Aipairs, 1);// check if AI has any matching cards
   }

   //check if you, AI, or the main decks are empty
   p = checkEmpty(playerDeck, AIDeck, AIDeck, 1, deck);

   if(p == 1) // if p =1 then one of them is and game over
   {
      break;
   }

   cout << "Here are your cards " << name << ": ";
   displayDeck(playerDeck); // Displays your cards


   aiAsks(playerDeck, AIDeck, deck, playerpairs, Aipairs, name);// Ai turns

   for(int i =0; i < 5; i++)
   {
      checkPair(playerDeck, playerpairs, 0); // check if you have any matching cards
      checkPair(AIDeck, Aipairs, 1); // check if Ai has any matching cards
   }

   cout << "Here are your cards " << name << ": ";
   displayDeck(playerDeck); // Displays your cards

    //check if you, AI, or the main decks are empty
   p = checkEmpty(playerDeck, AIDeck, AIDeck, 1, deck);

}while(p != 1);

int setYou = playerpairs->size();//counts the size of your matching sets of cards
int setAI = Aipairs->size();//counts the size of Ai's matching sets of cards

if(setYou > setAI)// if your set is larger then you win
{
   winsLose.push("Won");
   wins++;
```

```
            wltracker.pop();//Updates the winning and losing numbers in the queue
            wltracker.pop();
            wltracker.push(wins);
            wltracker.push(loses);
            cout << "You Win This Round!" << endl;
        }
        else if(setYou < setAI)// if your set is smaller then you lose
        {
            winsLose.push("Lost");
            loses++;
            wltracker.pop();//Updates the winning and losing numbers in the queue
            wltracker.pop();
            wltracker.push(wins);
            wltracker.push(loses);
            cout << "You Lost . . . To AI?" << endl;
        }
        else
        {
            winsLose.push("Draw"); // you don't win or lose with draw
            cout << "Its a draw!" << endl;
        }

        if(coins == 0)// if no more coins can't play again
        {
            cout << "No more coins, so no more plays! Sorry . . ." << endl;
            break;
        }

        cout << "Do you want to play Again?" << endl; // ask if they want to play again
        cin >> option;

        if(tolower(option[0]) == 'y')
        {
            cout << "New Round Created. . . " << endl;
        }

    }while(tolower(option[0]) == 'y');//if yes, the round restarts

    break;// break out of case
}
case 2:
{
    string option;// to see if player wants to playe again

    do
    {
        coins = coins - 20;//subtracts 20 from total coins
        displayCoins(mp,name,coins);//displays map with player name and coins
        cout << endl;

        cout << "Great! Game in session . . .  " << endl << endl;
```

```
gameRules(); // displays the rules of Go Fish

cout << endl;
cout << "========== GoFish! ==========" << endl;

list<int>* deck = new list<int>();//creates list for main deck
list<int>* playerDeck = new list<int>();// creates list for player deck
list<int>* AIDeck1 = new list<int>();//creates list for player AI 1 deck
list<int>* AIDeck2 = new list<int>();//creates list for player AI 2 deck

set<int>* playerpairs = new set<int>();//creates set for matching cards for player (you)
set<int>* Ai1pairs = new set<int>();//creates set for matching cards for AI 1
set<int>* Ai2pairs = new set<int>();//creates st for matching cards for AI 2

//Start Game
createDeck(deck);//puts cards in main deck
shuffle(deck);//shuffles main deck

for(int i = 0; i < 5; i++)
{
   playerDeck->push_back(disperse(deck)); //Give player Cards
}

for(int i = 0; i < 5; i++)
{
  AIDeck1->push_back(disperse(deck));  //Give AI Player 1 Cards
}

for(int i = 0; i < 5; i++)
{
  AIDeck2->push_back(disperse(deck));//Give AI Player 2 Cards
}

cout << "Here are your cards " << name << ": ";
displayDeck(playerDeck);// Displays player Deck

cout << endl;
checkPair(playerDeck, playerpairs, 0);//checks if player has 4 matching cards
checkPair(AIDeck1, Ai1pairs, 1);//checks if AI 1 has 4 matching cards
checkPair(AIDeck1, Ai2pairs, 2);//checks if Ai 2 has 4 matching cards
cout << endl;

int p = 0;// to hold if deck is empty

do
{
   cout << "******** Your Turn *********" << endl;
   //allows you ask any Ai player for a card
   askAIThree(playerDeck, AIDeck1, AIDeck2, deck, playerpairs, Ai1pairs, Ai2pairs);

   for(int i =0; i < 5; i++)
```

```
            {
               checkPair(playerDeck, playerpairs, 0); //checks if player has 4 matching cards
               checkPair(AIDeck1, Ai1pairs, 1);//checks if AI 1 has 4 matching cards
               checkPair(AIDeck2, Ai2pairs, 2);//checks if AI 2 has 4 matching cards
            }

            p = checkEmpty(playerDeck, AIDeck1, AIDeck2, 1, deck);//checks of any decks are
empty

            if(p == 1)// if any deck are empty, game over and break out of loop
            {
               break;
            }

            cout << "Here are your cards " << name << ": ";
            displayDeck(playerDeck);// displays your cards

            cout << "-----------------------------------------------" << endl;
            cout << "******** Player AI 1 Turn *********" << endl;
            //Ai 1's turn to ask anyone for card values
            aiAsksThree(playerDeck, AIDeck1, AIDeck2, deck, playerpairs, Ai1pairs, Ai2pairs, 1,
name);

            //Ai 2's turn to ask anyone for card values
            cout << "-----------------------------------------------" << endl;
            cout << "******** Player AI 2 Turn *********" << endl;
            aiAsksThree(playerDeck, AIDeck2, AIDeck1, deck, playerpairs, Ai2pairs, Ai1pairs, 2,
name);

            p = checkEmpty(playerDeck, AIDeck1, AIDeck2, 1, deck);//checks of any decks are
empty

         }while(p != 1); // end of any decks are empty

         int setYou = playerpairs->size();// size for matching cards for player
         int setAI1 = Ai1pairs->size();// size for matching cards for AI 1
         int setAI2 = Ai2pairs->size();// size for matching cards for AI 2

         if(setYou > setAI1 && setYou > setAI2)// if your set are bigger than AI 1 and AI 2, you win
         {
            winsLose.push("Won");
            wins++;
            wltracker.pop();//updates queue for wins and losses
            wltracker.pop();
            wltracker.push(wins);
            wltracker.push(loses);
            cout << "You Win This Round!" << endl;
         }
         else if(setYou < setAI1 && setAI2 < setAI1)// if AI 1 set are greater than you and AI 2, you
loose
         {
```

```
    winsLose.push("Lost");
    loses++;
    wltracker.pop();//updates queue for wins and losses
    wltracker.pop();
    wltracker.push(wins);
    wltracker.push(loses);
    cout << "Player AI 1 won this match!" << endl;
    cout << "You Lost To AI . . . ?" << endl;
}
else if(setYou < setAI2 && setAI1 < setAI2)//if Ai 2 set is greater, they win
{
    winsLose.push("Lost");
    loses++;
    wltracker.pop();//updates queue for wins and losses
    wltracker.pop();
    wltracker.push(wins);
    wltracker.push(loses);
    cout << "Player AI 2 won this match!" << endl;
    cout << "You Lost To AI . . . ?" << endl;
}
else if(setYou == setAI1)// tied with AI 1, then draw
{
    winsLose.push("Draw");
    cout << "You and Player AI 1 tied!" << endl;
}
else if(setYou == setAI2)// tied with AI 2 then draw
{
    winsLose.push("Draw");
    cout << "You and Player AI 2 tied!" << endl;
}
else if(setAI2 == setAI1)//AI 1 and AI 2 draw, you lose
{
    cout << "Player AI 1 and Player AI 2 tied!" << endl;
    winsLose.push("Lost");
    loses++;
    wltracker.pop();//updates queue for wins and losses
    wltracker.pop();
    wltracker.push(wins);
    wltracker.push(loses);
}
else{//everyone draws
    cout <<"Its a draw for everyone!" << endl;
}

if(coins == 0)// if no more coins left cant play again
{
    cout << "No more coins, so no more plays! Sorry . . ." << endl;
    break;
}

cout << "Do you want to play Again?" << endl;// if they want to play this mode again
```

```
            cin >> option;

        }while(tolower(option[0]) == 'y');// continue if yes

        break;
      }
      case 3:
      {
        //case 3 displays total wins and loses and outcome of each round in order
        cout << "================ Win/Lost History ==============" << endl;
        cout << "Total Wins: " << wltracker.front() << "     Total Loses: " << wltracker.back() << endl;
        cout << endl;
        cout << "---------- Track Each Round Outcome ---------" << endl;

        while(!winsLose.empty())
        {   //display stack
          cout << winsLose.top() << endl;
          winsLose.pop();
        }

        cout << "-----------------------------------------------" << endl << endl;

        break;
      }
      default:// if input not 1, 2, or 3
        cout << "Invalid input . . ." << endl;

    }

    if(coins == 0)// if no more coins then cant play anymore
    {
      cout << "You have exhausted your Coins . . ." << endl;
      break;
    }

    cout << "Would you like to go back to the menu?" << endl;// if they want to go back to menu
    cin >> menu;

  }while(tolower(menu[0]) == 'y'); //will go back if yes

  if(coins == 0)// if coins are zero it says Goodbye and exits
    {
      cout << "GOOD-BYE!" << endl;
    }
}
```

"GOFISH".cpp

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   GOFISH.cpp
 * Author: Yasmeen Allahaleh
 *
 *
 */

using namespace std;

/*
 *
 */
#include <cstdlib>
#include <iostream>
#include <ctime>
#include <iomanip>
#include <map>
#include <list>
#include <set>
#include <stack>
#include <queue>
#include <algorithm>
#include <iterator>

//Prototypes:
//Card Function
void createDeck(list<int> *deck);
void shuffle(list<int> *deck);
void displayDeck(list<int>* play);
int disperse(list<int> *cards);
int checkEmpty(list<int> *play, list<int> *opp, list<int> *opp2, int dis, list<int> *deck);
void gameRules();
//Go Fish
void addRemoveCards(list<int> *play, list<int> *opp, list<int> *deck, int value);
void askAI(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b);
void aiAsks(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b, string name);
void askAIThree(list<int> *play, list<int> *opp,list<int> *opp2, list<int> *deck, set<int> *a, set<int> *b,
set<int> *c);
void aiAsksThree(list<int> *play, list<int> *opp, list<int> *opp2, list<int> *deck, set<int> *a, set<int>
*b, set<int> *c, int d, string name);
void goFish(list<int> *play, list<int> *deck);
void checkPair(list<int> *play, set<int> *pair, int b);

/**
 * (For 2 player mode): allow you ask the Ai for value
 * if they have the cards they ask, it continues to be your turn until
```

```
 * they don't have a card you want
 * @param play- list for players current deck
 * @param opp- list for AI deck
 * @param deck - list main in deck
 * @param a - set for player (you) that hold matching cards
 * @param b - set for player AI that holds matching cards
 */
void askAI(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b)
{
   int empty = 0; // see if deck empty
   int value; // card you want
   bool found; // to see if card found in opposing player's deck

   do
   {
      cout << "     ******** Your Turn *********" << endl;
      cout << "-------------------------------------------------" << endl;
      cout << "Here are your Cards: " << endl;
      displayDeck(play);// displays your deck
      cout << "Asking AI for a value . . ." << endl;
      cout << "What value do you want to ask Player AI? : " << endl;
      cin >> value; // ask Ai for a value of the card

      cout << "AI is looking through His Cards . . ." << endl;

      list<int>::iterator findCard = find(opp->begin(), opp->end(), value); // iterator to find card
      found = (findCard != opp->end()); //returns if found card

      if(found)// if found
      {
         cout << "AI found the card you're looking for : 'Here you go . . .'" << endl;
         addRemoveCards(play, opp, deck, value); //allows the cards to be remove from opposing player
and into your deck

      }
      else
      {
         cout << "AI does not have the card you want : 'Too Bad . . .GO FISH!'" << endl;
         goFish(play, deck);// You have to draw from Deck
      }

      cout << endl;

      checkPair(play, a, 0);// check if you have 4 matching cards
      checkPair(opp, b, 1);// check if AI have 4 matching cards

      empty = checkEmpty(play, opp, opp, 0, deck);// check if decks are empty

   }while(found && empty == 0);// continue to be your turn until deck empty or card value not found

}
```

```
/**
 * Add cards found in opposing player's deck into yours and remove it from theirs
 * @param play- list of player's deck who asked for value
 * @param opp - list of player who was asked deck
 * @param deck - list main deck
 * @param value - value that player asked for
 */
 void addRemoveCards(list<int> *play, list<int> *opp, list<int> *deck, int value)
 {

   int countCard = count(opp->begin(), opp->end(), value);// counts how many cards they have of asked
value

   cout << "He had : " << countCard << " card(s) with the value of: "
        << value << endl; // displays ho many cards he had of that value

   for(int i=0; i < 52; i++)
   {
      opp->remove(value);// remove it from opposing deck
   }

   for(int i=0; i < countCard; i++)
   {
      play->push_back(value);// adds them to player who asked deck's
   }

 }

 /**
  * for 2 Player Mode
  * Allows the Ai to ask your for the card they want
  * If you have the value they want its their turn again until they ask for value
  * you don't have or deck is empty
  * @param play - list of the player (your)'s current deck of cards
 * @param play- list for players current deck
 * @param opp- list for AI deck
 * @param deck - list main in deck
 * @param a - set for player (you) that hold matching cards
 * @param b - set for player AI that holds matching cards
  * @param name - the name of the player
  */
 void aiAsks(list<int> *play, list<int> *opp, list<int> *deck, set<int> *a, set<int> *b, string name)
 {

   int empty = 0;// hold if any deck is empty
   bool found = true; // if balue of card was found

   do
   {
      cout << "     ******** AI's Turn *********" << endl;
```

```
        cout << "-----------------------------------------------" << endl;
        int card = opp->back();// asks for the card in the back of their deck

        cout << "Hey " << name << ", do you have a : " << card << endl; // asks if you have the certain card

        list<int>::iterator findCard = find(play->begin(), play->end(), card);// iterates through your deck
        found = (findCard != play->end()); // holds if card was found

        if(found)// if the card was found in your deck
        {
           cout << "You found the card he was looking for : 'Here you go . . .'" << endl;
           addRemoveCards(opp, play, deck, card);// adds the cards to the AI's deck and removes them from
yours

        }
        else
        {
           cout << "You do not have that card : 'Too Bad . . .GO FISH!'" << endl;
           goFish(opp, deck);// if card not found in your deck, Ai pulls from main deck
        }

        cout << endl << endl;

        checkPair(play, a, 0);//check if you have any 4 matching cards
        checkPair(opp, b, 1);// checks if Ai has any 4 matching cards

        empty = checkEmpty(play, opp, opp, 0, deck);// checks if any decks are empty

    }while(found && empty == 0);  // counties if they asks you for card you have and decks aren't empty


    cout << endl;


}

/**
 * Allows the player to pull from main deck
 * @param play - the player's list of their deck who pulls from deck
 * @param deck - the main deck that the card gets pulled from
 */
void goFish(list<int> *play, list<int> *deck)
{
    cout << "Pulling card from deck. . ." << endl;
    int pull = disperse(deck);//card pulled from deck

    play->push_back(pull);// puts it into the players deck
}

/**
 * Checks if player has 4 matching cards in their deck
```

```
 * @param play - the list of player deck that's being checked
 * @param pair - the set that hold matching cards
 * @param b - number that indicates which player is being checked
 */
void checkPair(list<int> *play, set<int> *pair, int b)
{

    int countCard = 0;//counts the number cards for a certain value
    int card = 0;

    for(auto i : *play)// goes through list to find number of cards
    {
      countCard = count(play->begin(), play->end(), i);//goes through and counts how many cards their are
for each card
      if(countCard == 4)
      {
         card = i;// assign the cards value found with 4 matching cards

      }
    }

    if (card != 0)// if no card found
    {
      pair->insert(card);// insert the value of card into set
      play->remove(card); //removes from current player's deck
    }

    if(b == 0 && card != 0)//if your deck contained 4 matching cards
    {
      cout << endl;
      cout << "You found a set of " << card << " in your Deck!" << endl;
      cout << endl;
    }
    if(b == 1 && card != 0)//if AI 1's deck contained 4 matching cards
    {
      cout << endl;
      cout << "Player AI 1 found a set of " << card << " in his Deck!" << endl;
      cout << endl;
    }
    if(b == 2 && card != 0)//if AI 2's deck contained 4 matching cards
    {
      cout << endl;
      cout << "Player AI 2 found a set of " << card << " in his Deck!" << endl;
      cout << endl;
    }
}
```

"CardFunction".cpp

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   CardFunction.cpp
 * Author: Yasmeen Allahaleh
 *
 *
 */


#include <cstdlib>
#include <iostream>
#include <ctime>
#include <iomanip>
#include <map>
#include <list>
#include <set>
#include <stack>
#include <queue>
#include <algorithm>
#include <iterator>


using namespace std;

//Prototypes:
void createDeck(list<int> *deck);
void shuffle(list<int> *deck);
void displayDeck(list<int>* play);
int disperse(list<int> *cards);
int checkEmpty(list<int> *play, list<int> *opp, list<int> *opp2, int dis, list<int> *deck);
void gameRules();
void displayCoins(map<string, int> a, string name, int coins);

/**
 * Putting the cards in the main deck
 * @param deck- the list of the main deck
 */
void createDeck(list<int> *deck)
{
   int cardNum[] = {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}; // the cards being into the main deck

   copy(begin(cardNum), end(cardNum), back_inserter(*deck));// copy the values into deck

}
```

```
/**
 * Shuffles the main deck
 * @param deck - the list of the main deck
 */
void shuffle(list<int> *deck)
{
   int p1, p2;// hold the values of cards

   for(int i = 0; i < 1000; i++)// iterates to shuffle
   {
     p1 = rand() % deck->size();// picks a position to swap
     p2 = rand() % deck->size();// picks a position to swap

     list<int>::iterator pt1 = deck->begin();//iterates to the card
     advance(pt1, p1);
     list<int>::iterator pt2 =deck->begin();//iterates to the card
     advance(pt2, p2);

     swap(*pt1, *pt2);//swaps the two cards

     if(i>= 2){
         break;
      }

   }
}

/**
 * Displays the deck
 * @param play - the list of the player's deck
 */
void displayDeck(list<int> * play)
{
   for(auto i : *play)
   {
      cout << i << " ";
   }
   cout << endl;
}

/**
 * deals a card to put in a player's deck
 * @param cards - the list of the main deck
 * @return the value of the card
 */
int disperse(list<int> *cards)
{
   int num;

   num = cards->back();//pull from the back of the deck
```

```
    cards->pop_back();//remove it from main deck

    return num;//return the value of card
}

/**
 * Check of any deck is empty
 * @param play - the list of the player's deck
 * @param opp - the list of the opposing AI deck
 * @param dis - indicates if message should be displayed
 * @param deck - the list of the main deck
 * @return value 1 if any deck is empty
 */
int checkEmpty(list<int> *play, list<int> *opp, list<int> *opp2, int dis, list<int> *deck)
{
    int value = 0;// zero mean no deck is empty

    if(play->size() == 0){// your deck has a size of zero, then its empty and game is over

        if(dis == 1)
        {
            cout << endl;
            cout << "*========================*" << endl;
            cout << "    Your deck is empty!" << endl;
            cout << ". . . Game is over  . . ." << endl;
            cout << "*========================*" << endl << endl;
        }

        value = 1;// 1 is returned to indicate that its empty
        return value;
    }
    else if(opp->size() == 0){// AI 1 deck has a size of zero, then its empty and game is over

        if(dis == 1)
        {
            cout << endl;
            cout << "*========================*" << endl;
            cout << "    AI 1's deck is empty!" << endl;
            cout << ". . . Game is over  . . ." << endl;
            cout << "*========================*" << endl << endl;
        }

        value = 1;// 1 is returned to indicate that its empty
        return value;
    }
    else if(opp2->size() == 0){// AI 2 deck has a size of zero, then its empty and game is over

        if(dis == 1)
        {
            cout << endl;
            cout << "*========================*" << endl;
```

```
            cout << " AI 2's deck is empty!" << endl;
            cout << ". . . Game is over  . . ." << endl;
            cout << "*========================*" << endl <<endl;
        }

        value = 1;// 1 is returned to indicate that its empty
        return value;
    }
    else if(deck->size() == 0){// AI 2 deck has a size of zero, then its empty and game is over

        if(dis == 1)
        {
            cout << endl;
            cout << "*========================*" << endl;
            cout << " The main deck is empty!" << endl;
            cout << ". . . Game is over  . . ." << endl;
            cout << "*========================*" << endl <<endl;
        }

        value = 1;// 1 is returned to indicate that its empty
        return value;
    }

    return value;
}

/**
 * Describes the rules for  the card game GoFish
 */
void gameRules()
{
    cout << "====================== Game Rules ======================" << endl;
    cout << "* Note: This game is played against an AI player " << endl;
    cout << "  that will simulate a real person playing" << endl;
    cout << "Objective: To win get the most pairs of the cards" << endl;
    cout << "before your, the opposing player, or the middle " << endl;
    cout << "deck runs out." << endl;
    cout << "1.) You and the AI player are dealt 5 cards" << endl;
    cout << "    from the shuffled deck." << endl;
    cout << "2.) The first player will ask if the opposing player" << endl;
    cout << "    has a certain value" << endl;
    cout << "3.) If the opposing player that value, then they must " << endl;
    cout << "    give all of their cards with the asked value" << endl;
    cout << "4.) When this happens, your get to go again and ask" << endl;
    cout << "    for another card" << endl;
    cout << "5.) If they do not have the value, they call GO FISH!" << endl;
    cout << "    and you must pull from the middle deck and then" << endl;
    cout << "    it's the opposing player's turn" << endl;
    cout << "6.) The same goes whenever it's the opposing player's " << endl;
    cout << "    turn to ask" << endl;
    cout << "7.) If you pull card from the deck or receive them from " << endl;
```

```
    cout << "    from the opposing player that match in a total of 4" << endl;
    cout << "    cards, then you put them to the side as you have a " << endl;
    cout << "    a set of those values. " << endl;
    cout << "*REMEMBER* The more sets you have the better! " << endl;
    cout << "8.) When your cards, the opposing player's cards, or the" << endl;
    cout << "    the deck finishes, the game ends and whoever has the " << endl;
    cout << "    most sets of matching cards wins the game!" << endl;
    cout << "IMPORTANT: If you tie against an AI nobody wins. If two" << endl;
    cout << "           AIs tie, you loses." << endl;
    cout << "============================================================" << endl;

}

/**
 * display the player's inserted name and amount of coins available
 * @param a - the map
 * @param name - the name that the player inputted
 * @param coins - the amount of coins the player has currently
 */
void displayCoins(map<string, int> a, string name, int coins)
{
    a[name] = coins;

    map<string, int>::iterator c = a.begin();//

    while (c != a.end()) {
        cout << "Player Name: " << c->first
             << ", Coins: " << c->second << endl;
          ++c;
        } // display the map
}

int rec(int m, int n, list<int> *play, list<int> *deck)
{
    if(m == n)
    {

        return 0;
    }

    play->push_back(disperse(deck));
    return rec(m+1,n,play,deck);;

}

void recSort(list<int> *play, int n){

    int track = 0;


    for(int i = 0; i < n-1; i++){
```

```
        play->sort();
    }

    if(track == 0)
        return;

    recSort(play, n-1);

}
```