Project 2

<Monopoly>

CIS-17A 47538
Name: Abu-Ghazaleh, Zayd
Date: 12/1/2024

Introduction

Title: Monopoly

For this project, I created monopoly.
Monopoly is a game that is explained simply by the name, as the players fight to gain monopoly of land. Whoever is able to gain monopoly of all railroads or an entire row is the winner.
Each player rolls two dice per round, moving onto both safe or dangerous spots.

Once one player owns a piece of land, any player that moves onto this land

must pay the toll. Through a mixture of strategy and luck, there is one winner.

This game creates a fun experience for those playing, creating competition and getting the players used to making careful decisions with money. This game is able to teach both the young and old about budgeting and investment in a fun and competitive way.

Summary

Project size: about 1233 lines
The number of variables: 23 (including classes, not those inside the classes)
The number of functions: 6

The number of classes:4

This project demonstrates almost every concept that we have learned so far.
Over the iterations, I spent considerable time modifying the program to allow for classes and simpler, more understandable code. The part that ended up taking the longest time was simply fixing bugs that stemmed from changes in the code. This alone took me over 30 hours to fix.

This project took me about 2 months overall, as this project was a lot more complex than I believed, and even more complex to change.
I ran into countless errors that took hefty amounts of time to fix.
I'm not completely happy with how the project ended up, as I wanted to add more players, but I am satisfied with what I have now.
I've realized that C++ has both its pros and cons, and it has been fun figuring that out.
Although this game was simple in theory, I ended up utilizing very complex concepts that we have learned in the class.

Description

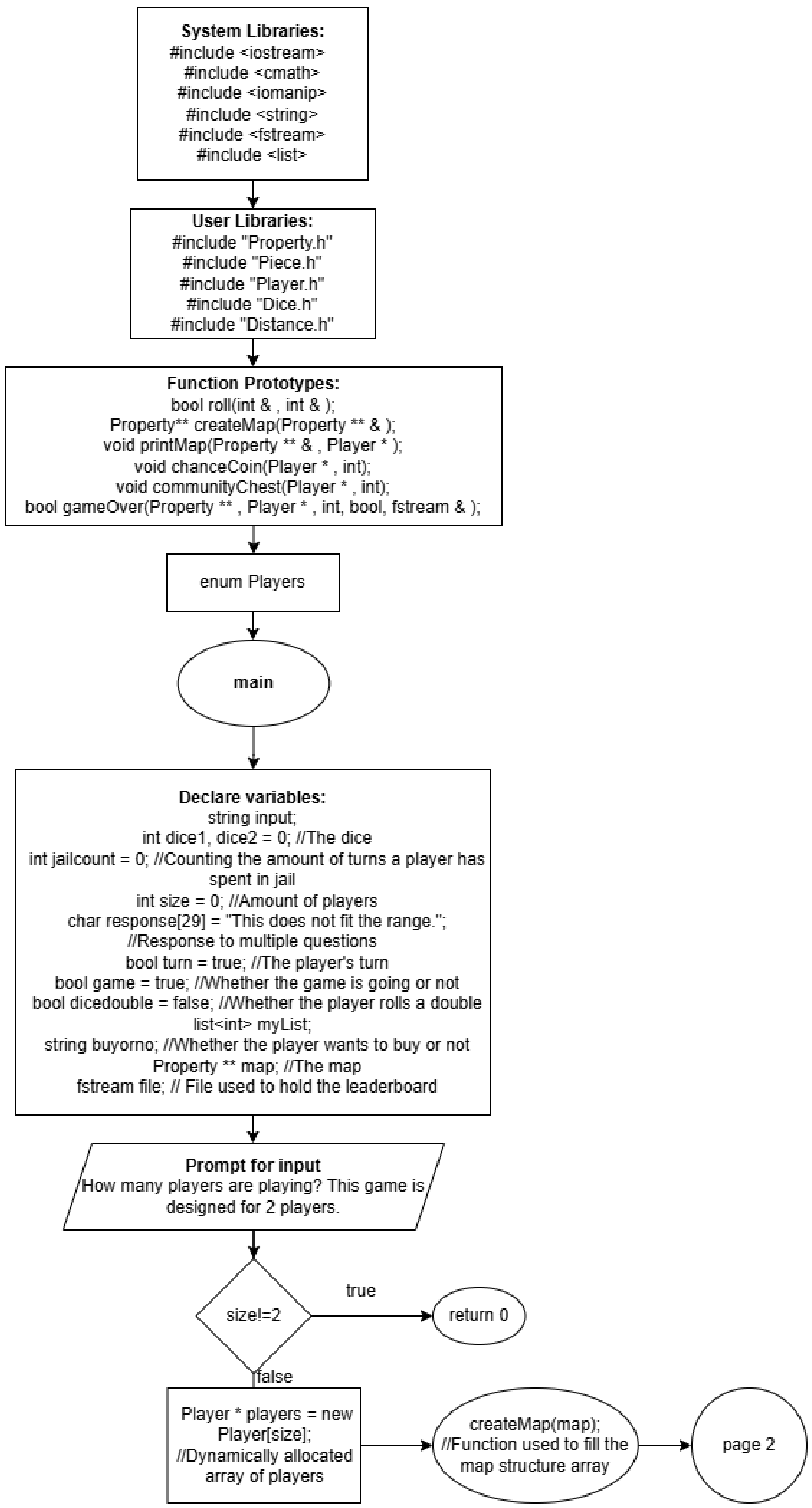
The main point of this program was to effectively and efficiently recreate monopoly in C++. I wanted this program to be as fun, engaging, and immersive as the real monopoly, so I spent a lot of time creating the print function which outputs a real map for the players.

Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
13		Classes			
	1 to 3	Instance of a Class	32,34,36,55	4	Header files containing classes
	4	Private Data Members	32,34,36,55	4	Header files containing private data members
	5	Specification vs. Implementation	36	4	#include header, cpp included in folder
	6	Inline	32,34,36,55	4	Inline included in header files
	7, 8, 10	Constructors	36	4	Lines 9-19 in header file
	9	Destructors	36	4	Line 53 in header file
	12	Arrays of Objects	79	4	
	16	UML	40-53	4	Created in text
14		More about Classes			
	1	Static	36	5	Line 7 in header file
	2	Friends		2	
	4	Copy Constructors	36	5	Line 20 in header file
	5	Operator Overloading	Lines 150-177, 244	8	In header file, utilized in lines on left
	7	Aggregation	34	6	In header file, piece class aggregated into player class
15		Inheritance			
	1	Protected members	36	6	Line 4 in header file
	2 to 5	Base Class to Derived	32,34,36,55	6	In header files
	6	Polymorphic associations	36	6	Line 24 in header file
	7	Abstract Classes	55	6	Line 6 in header file
16		Advanced Classes			
	1	Exceptions	36	6	Line 26 in header
	2 to 4	Templates	55	6	Line 8 in header
	5	STL	86,98	6	
		Sum		100	

Flow Chart (Next page)


System Libraries:

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <string>
#include <fstream>
#include <list>
```

User Libraries:

```
#include "Property.h"
#include "Piece.h"
#include "Player.h"
#include "Dice.h"
#include "Distance.h"
```

Function Prototypes:

```
bool roll(int & , int & );
Property** createMap(Property ** & );
void printMap(Property ** & , Player * );
void chanceCoin(Player * , int);
void communityChest(Player * , int);
bool gameOver(Property ** , Player * , int, bool, fstream & );
```

```
enum Players
```

Declare variables:

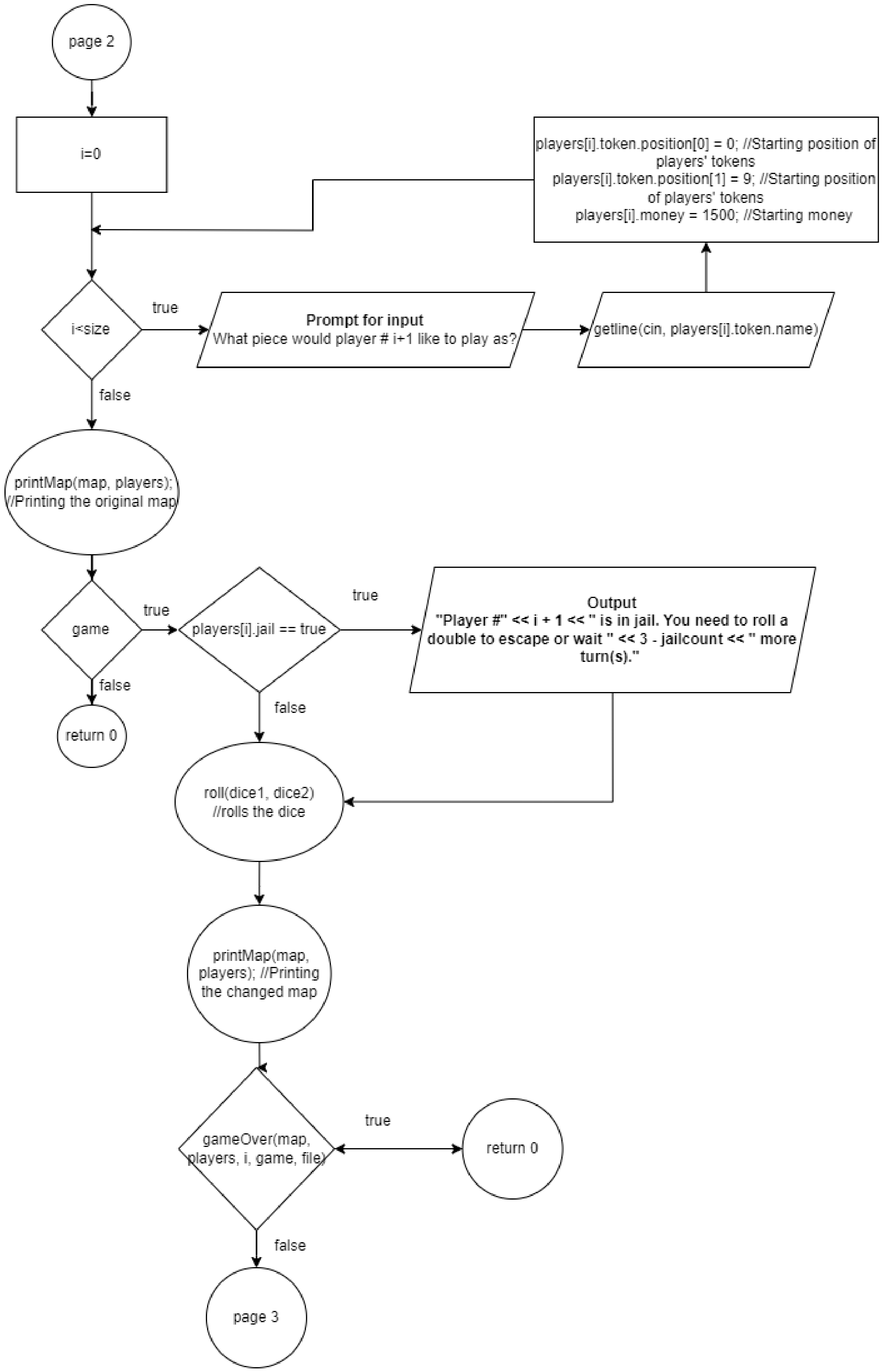
```
string input;
int dice1, dice2 = 0; //The dice
int jailcount = 0; //Counting the amount of turns a player has spent in jail
int size = 0; //Amount of players
char response[29] = "This does not fit the range.";
//Response to multiple questions
bool turn = true; //The player's turn
bool game = true; //Whether the game is going or not
bool dicedouble = false; //Whether the player rolls a double
list<int> myList;
string buyorno; //Whether the player wants to buy or not
Property ** map; //The map
fstream file; // File used to hold the leaderboard
```

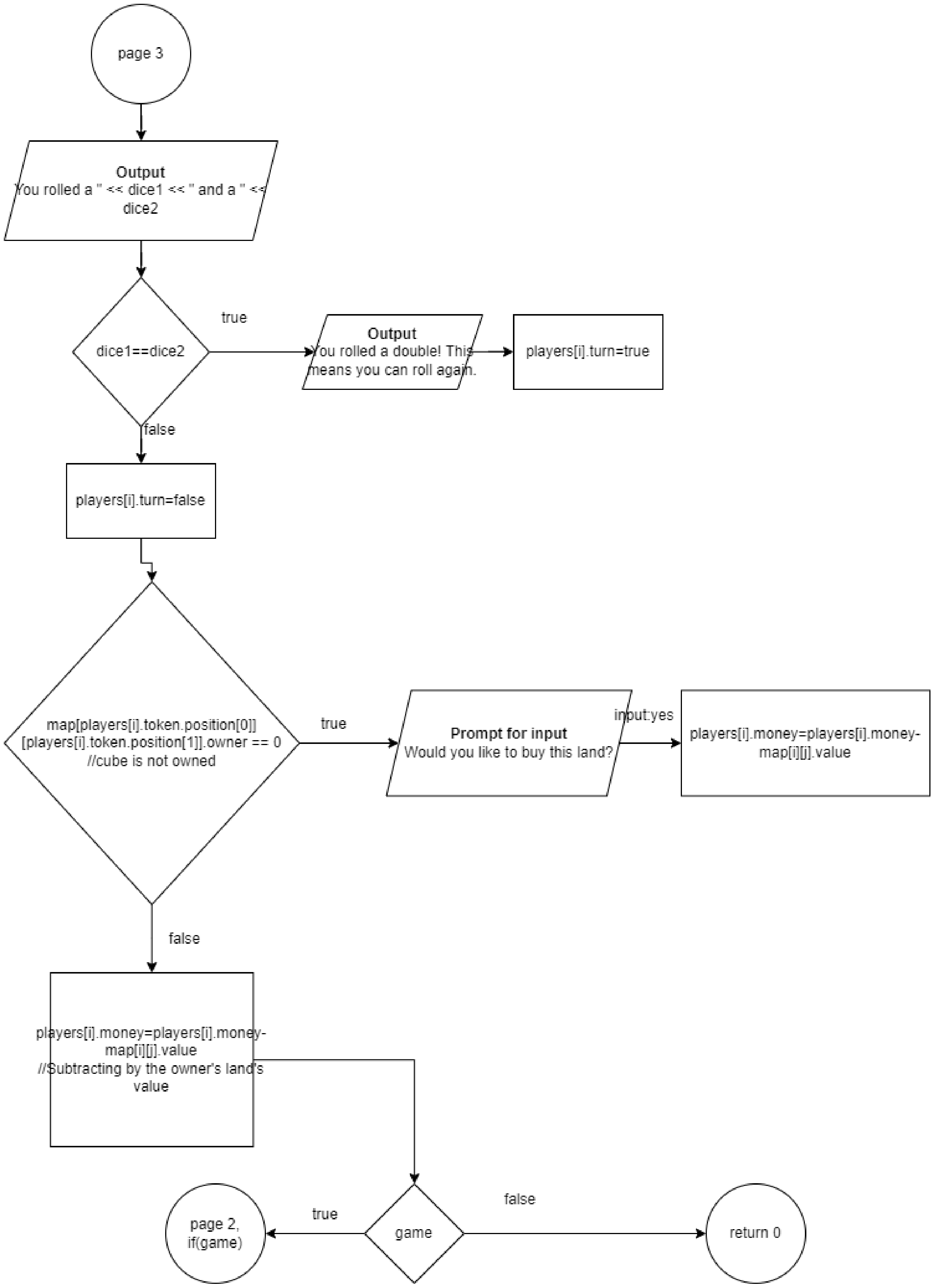
Prompt for input
How many players are playing? This game is designed for 2 players.

```
size!=2
```

```
Player * players = new Player[size];
//Dynamically allocated array of players
```

```
createMap(map);
//Function used to fill the map structure array
```


page 2
i=0
players[i].token.position[0] = 0; //Starting position of players' tokens
players[i].token.position[1] = 9; //Starting position of players' tokens
players[i].money = 1500; //Starting money
i<size
true
Prompt for input
What piece would player # i+1 like to play as?
getline(cin, players[i].token.name)
false
printMap(map, players); //Printing the original map
game
true
players[i].jail == true
true
Output
"Player #" << i + 1 << " is in jail. You need to roll a double to escape or wait " << 3 - jailcount << " more turn(s)."
false
return 0
false
roll(dice1, dice2) //rolls the dice
printMap(map, players); //Printing the changed map
gameOver(map, players, i, game, file)
true
return 0
false
page 3


page 3
Output
You rolled a " << dice1 << " and a " << dice2
dice1==dice2
true
Output
You rolled a double! This means you can roll again.
players[i].turn=true
false
players[i].turn=false
map[players[i].token.position[0]][players[i].token.position[1]].owner == 0 //cube is not owned
true
Prompt for input
Would you like to buy this land?
input:yes
players[i].money=players[i].money-map[i][j].value
false
players[i].money=players[i].money-map[i][j].value //Subtracting by the owner's land's value
game
true
page 2, if(game)
false
return 0

Pseudo Code

```
Initialize

Output how many players are playing
If players does not equal two
    Return 0

Loop for the amount of players
     Ask the player the name of the piece and fill the player structure array with information

printMap(map, players)

While the game boolean is true

    roll(dice1, dice2)

            If  the roll is a double

                        Player's turn is true

            Else

                        Player's turn is false

            Move the player's piece based on the dice

            printMap(map, players)

            game= gameOver(map, players, I, game, file)

If game is true

            If player is on a buyable piece of land

                        Output if the player would like to buy

                        Subtract that amount from the player's money

            Else if the player is on an owned piece of land

                        Subtract that amount from the player's money

            Else if the player is on a chance coin or community chest
```

```
        Pull a card/roll a coin
    Else
    Move onto the next player
Else
    Return 0;
```

Major Variables

Type	Variable Name	Description	Location
Integer	jailcount	Counts the amount of turns player has spent in jail	main(int argc, char ** argv)
	Size	The amount of players	main(int argc, char ** argv)
	len	Length of strings for printing	printMap(Property ** & map, Player * players)
	chance	The chance coin that you rolled	chanceCoin(Player * players, int i)
	chest	The card you pulled from the chest	communityChest(Player * players, int i)
	ownercount	Used to count whether a player owns an entire row	gameOver(Property ** map, Player * players, int i, bool game, fstream & file)
	ownercount2	Used to count whether a player owns an entire row	gameOver(Property ** map, Player * players, int i, bool game, fstream & file)
	railroadcount	Used to count whether a player owns all the railroads	gameOver(Property ** map, Player * players, int i, bool game, fstream & file)
bool	turn	The player's turn	main(int argc, char ** argv)
	game	Whether the game is going or not	main(int argc, char ** argv)
	dicedouble	Whether the player rolls a double	main(int argc, char ** argv)
	pos	Whether the player is on this square	printMap(Property ** & map, Player * players)

string	input	Used to get the player’s input when rolling	main(int argc, char ** argv)
	buyorno	Whether the player wants to buy or not	main(int argc, char ** argv)
	leng	Used to convert an integer to a string and find the length of the string	printMap(Property ** & map, Player * players)

	answer	Used to get the player’s input when rolling a coin or pulling a card	communityChest(Player * players, int i)
			&&
			chanceCoin(Player * players, int i)

	winner	Holds the name of the winner	gameOver(Property ** map, Player * players, int i, bool game, fstream & file)
C-String	response	Response to multiple questions	main(int argc, char ** argv)
Property **	Map	Structure array used to hold values of the map	main(int argc, char ** argv)
Player *	players	Structure array used to hold the players	main(int argc, char ** argv)
fstream	file	File used to hold the leaderboard	main(int argc, char ** argv)
list	myList	List to hold players	main(int argc, char ** argv)
Dice	diceroll[2]	Object array to hold dice	main(int argc, char ** argv)

Reference

1. Textbook
2. Lectures
3. Monopoly Board

Program

```
/*

 * File:   main.cpp

 * Author: Zayd Abu-Ghazaleh

 * Created on September 9th, 10:24 AM

 * Purpose:  Template which is to be copied for all future
 *          Homework, Labs, Projects, Test, etc...
 */

//System Libraries
#include <iostream>  //I/O Library

#include <cmath>

#include <iomanip>

#include <string>

#include <fstream>

#include <list>

using namespace std;

//User Libraries
#include "Property.h"

#include "Piece.h"
```

```
#include "Player.h"

#include "Dice.h"

int Dice::rolls=0;
/*
|--------------------------------------|
|                UML                   |
|--------------------------------------|
|                Dice                  |
|--------------------------------------|
|            - value : int             |
|         - rolls : static int         |
|--------------------------------------|
|            +Dice() : Dice            |
|        +Dice(int input) : Dice       |
|        +Dice(Dice &obj) : Dice       |
|          +getValue() : int           |
| setValue(int inputval| : virtual void |
|--------------------------------------| */

#include "Distance.h"
```

```
enum Players {
  PLAYER1,
  PLAYER2
};

//Global Constants Only
//Well known Science, Mathematical and Laboratory Constants
bool roll(Dice[]);
Property** createMap(Property ** & );
void printMap(Property ** & , Player * );
void chanceCoin(Player * , int);
void communityChest(Player * , int);
bool gameOver(Property ** , Player * , int, bool, fstream & );
//Function Prototypes

//Execution of Code Begins Here

int main(int argc, char ** argv) {
  //Set the random number seed here
  string input;
  srand(static_cast < int > (time(NULL)));
  //Declare all variables for this function
  Dice diceroll[2];
```

```
int jailcount[2] = {0,0}; //Counting the amount of turns a player has spent in jail

int size = 0; //Amount of players

char response[29] = "This does not fit the range."; //Response to multiple questions

bool turn = true; //The player's turn

bool game = true; //Whether the game is going or not

bool dicedouble = false; //Whether the player rolls a double

list<int> myList;

string buyorno; //Whether the player wants to buy or not

Property ** map; //The map

createMap(map); //Function used to fill the map structure array

fstream file; // File used to hold the leaderboard

cout << "How many players are playing? This game is designed for 2 players." << endl;

cin >> size; //Amount of players

if (size != 2) {

  cout << response << endl;

  return 0;

}

else{

    myList.push_back(2);

}

cin.ignore();

Player * players = new Player[size]; //Dynamically allocated array of players
```

```
  for (int i = 0; i < size; i++) { //For loop that fills the players array with information

    cout << "What piece would player #" << i + 1 << " like to play as?" << endl;

    cout << "Enter any object you would like, (1-10 letters), this will be used as a piece." << endl;

    getline(cin, players[i].token.name);

    if (players[i].token.name.length() < 1 || players[i].token.name.length() > 10) {

      cout << response << endl;

      return 0;

    }


    players[i].token.position[0] = 0; //Starting position of players' tokens

    players[i].token.position[1] = 9; //Starting position of players' tokens

    players[i].money = 1500; //Starting money

  }


  printMap(map, players); //Printing the original map

  while (game) { //While game is continuing


    for (int i = 0; i < size; i++) { //For loop for each player


      players[i].turn = true; //Sets players turn to true

      while (players[i].turn) {

        if (players[i].jail == true) { //Checks for jail

          cout << "Player #" << i + 1 << " is in jail. You need to roll a double to escape or wait " << 3 -
jailcount[i] << " more turn(s)." << endl;
```

```
}
cout << "Is player #" << i + 1 << " ready to roll? Type anything to roll." << endl;
getline(cin, input);
if (roll(diceroll) == true) { //If the roll function returns true, player rolled a double
  dicedouble = true;
} else {
  dicedouble = false;
}
if (players[i].jail == true) { //Goes through conditions of escaping jail
  if (dicedouble == true) {
    players[i].jail == false;
  } else if (jailcount[i] >= 2) {
    players[i].jail == false;
  } else {
    jailcount[i]++;
  }
}

players[i].turn = dicedouble; //Sets turn to whether player rolled a double or not
if (players[i].jail == true) {


} else {
```

```
          if (players[i].token.position[0] == 0) { //This entire if, else if, else statement moves the player's
piece
            if (players[i].token.position[1] - (diceroll[0] + diceroll[1]) < 0) {
              players[i].token.position[0] = (diceroll[0] + diceroll[1]) - players[i].token.position[1];
              players[i].token.position[1] = 0;
            } else {
              players[i].token.position[1] = players[i].token.position[1] - (diceroll[0] + diceroll[1]);
            }
          } else if (players[i].token.position[0] == 9) {
            if (players[i].token.position[1] + (diceroll[0] + diceroll[1]) > 9) {
              players[i].token.position[0] = 9 - ((diceroll[0] + diceroll[1]) - (9 -
players[i].token.position[1]));
              players[i].token.position[1] = 9;
            } else {
              players[i].token.position[1] = players[i].token.position[1] + (diceroll[0] + diceroll[1]);
            }
          } else {
            if (players[i].token.position[1] == 9) {
              if (players[i].token.position[0] - (diceroll[0] + diceroll[1]) < 0) {
                players[i].token.position[1] = 9 - ((diceroll[0] + diceroll[1]) - players[i].token.position[0]);
                players[i].token.position[0] = 0;
              } else {
                players[i].token.position[0] = players[i].token.position[0] - (diceroll[0] + diceroll[1]);
              }
            } else {
              if (players[i].token.position[0] + (diceroll[0] + diceroll[1]) > 9) {
                players[i].token.position[1] = (diceroll[0] + diceroll[1]) - (9 - players[i].token.position[0]);
                players[i].token.position[0] = 9;
              } else {
                players[i].token.position[0] = players[i].token.position[0] + (diceroll[0] + diceroll[1]);
              }
            }


          }
        }
        printMap(map, players); //Prints map again
        game = gameOver(map, players, i, game, file); //Checks whether game is over
        if (game) {
          cout << "You rolled a " << diceroll[0].getValue() << " and a " << diceroll[1].getValue() <<
endl;
          if (dicedouble == true) cout << "You rolled a double! This means you can roll again." << endl;
          if (map[players[i].token.position[0]][players[i].token.position[1]].owner == 0) {
            if (map[players[i].token.position[0]][players[i].token.position[1]].type != "Government" &&
map[players[i].token.position[0]][players[i].token.position[1]].type != "Coin" &&
map[players[i].token.position[0]][players[i].token.position[1]].type != "Chest") {
              cout << "Would you like to buy " <<
map[players[i].token.position[0]][players[i].token.position[1]].name << " for $" <<
map[players[i].token.position[0]][players[i].token.position[1]].value << "? Type 'yes' to buy and 'no'
to not buy." << endl;
              getline(cin, buyorno);
              while (buyorno != "yes" && buyorno != "no") {
                cout << "Enter 'yes' or 'no'" << endl;
                getline(cin, buyorno);
              }
              if (buyorno == "yes") {
                if (players[i].money <
map[players[i].token.position[0]][players[i].token.position[1]].value) {
                cout << "You do not have enough money!" << endl;
                } else {
                map[players[i].token.position[0]][players[i].token.position[1]].owner = i + 1;
                players[i].money = players[i].money -
map[players[i].token.position[0]][players[i].token.position[1]].value;
                cout << "Player #" << i + 1 << " now has $" << players[i].money << endl;
                }
              }
            } else {
              if (map[players[i].token.position[0]][players[i].token.position[1]].type == "Coin") {
                chanceCoin(players, i);
              } else if (map[players[i].token.position[0]][players[i].token.position[1]].type == "Chest") {
                communityChest(players, i);
              }
            }
          } else {
            if (map[players[i].token.position[0]][players[i].token.position[1]].owner == i + 1) {

              cout << "You own the property " <<
map[players[i].token.position[0]][players[i].token.position[1]].name << endl;

            } else {

              cout << "The property " <<
map[players[i].token.position[0]][players[i].token.position[1]].name << " is owned by player " <<
map[players[i].token.position[0]][players[i].token.position[1]].owner << "." << endl;

              cout << "You owe $" <<
map[players[i].token.position[0]][players[i].token.position[1]].value << endl;



            }

          }

        } else {

          return 0;

        }

      }

    }


   }




   for (int i = 0; i < 10; i++) {

     delete map[i];

   }

   delete map;
```

```
  return 0;

}


//Function Implementations


bool roll(Dice diceroll[]) {

  bool turn;

  diceroll[0].setValue(rand() % 4 + 1);

  diceroll[1].setValue(rand() % 4 + 1);


  if (diceroll[0]==diceroll[1]) turn = true;

  else turn = false;


  return turn;

}

Property** createMap(Property ** & map) { //This function creates the entire map in a structure
array

  map = new Property * [10];

  for (int i = 0; i < 10; i++) {

    map[i] = new Property[10];

  }


  for (int i = 0; i < 10; i++) {

    for (int j = 0; j < 10; j++) {
```

```
if (i == 0) {
  if (j == 0) {
    map[i][j].name = "JAIL";
    map[i][j].type = "Government";
    map[i][j].value = 0;
    map[i][j].owner = 0;

  } else if (j == 1) {
    map[i][j].name = "CONNECT. AVE";
    map[i][j].type = "Blue";
    map[i][j].value = 120;
    map[i][j].owner = 0;

  } else if (j == 2) {
    map[i][j].name = "VERMONT AVE";
    map[i][j].type = "Blue";
    map[i][j].value = 100;
    map[i][j].owner = 0;

  } else if (j == 3) {
    map[i][j].name = "CHANCE";
    map[i][j].type = "Coin";
    map[i][j].value = 0;
  map[i][j].owner = 0;

} else if (j == 4) {
  map[i][j].name = "ORIENTAL AVE";
  map[i][j].type = "Blue";
  map[i][j].value = 100;
  map[i][j].owner = 0;

} else if (j == 5) {
  map[i][j].name = "VAN RAILROAD";
  map[i][j].type = "Railroad";
  map[i][j].value = 200;
  map[i][j].owner = 0;

} else if (j == 6) {
  map[i][j].name = "BALTIC AVE";
  map[i][j].type = "Brown";
  map[i][j].value = 60;
  map[i][j].owner = 0;

} else if (j == 7) {
  map[i][j].name = "COMM. CHEST";
  map[i][j].type = "Chest";
```

```
      map[i][j].value = 0;
      map[i][j].owner = 0;

    } else if (j == 8) {
      map[i][j].name = "MED. AVE";
      map[i][j].type = "Brown";
      map[i][j].value = 60;
      map[i][j].owner = 0;

    } else if (j == 9) {
      map[i][j].name = "GO";
      map[i][j].type = "Government";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    }
  } else if (i == 9) {
    if (j == 0) {
      map[i][j].name = "FREE PARKING";
      map[i][j].type = "Government";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    } else if (j == 1) {
      map[i][j].name = "KENTUCKY AVE";
  map[i][j].type = "Red";

  map[i][j].value = 220;

  map[i][j].owner = 0;

} else if (j == 2) {

  map[i][j].name = "CHANCE";

  map[i][j].type = "Coin";

  map[i][j].value = 0;

  map[i][j].owner = 0;


} else if (j == 3) {

  map[i][j].name = "INDIANA AVE";

  map[i][j].type = "Red";

  map[i][j].value = 220;

  map[i][j].owner = 0;


} else if (j == 4) {

  map[i][j].name = "ILLINOIS AVE";

  map[i][j].type = "Red";

  map[i][j].value = 240;

  map[i][j].owner = 0;


} else if (j == 5) {

  map[i][j].name = "B&O RAILROAD";
  map[i][j].type = "Railroad";

  map[i][j].value = 200;

  map[i][j].owner = 0;


} else if (j == 6) {

  map[i][j].name = "ATLANTIC AVE";

  map[i][j].type = "Yellow";

  map[i][j].value = 260;

  map[i][j].owner = 0;


} else if (j == 7) {

  map[i][j].name = "VENTNOR AVE";

  map[i][j].type = "Yellow";

  map[i][j].value = 260;

  map[i][j].owner = 0;


} else if (j == 8) {

  map[i][j].name = "MARV GARDENS";

  map[i][j].type = "Yellow";

  map[i][j].value = 280;

  map[i][j].owner = 0;


} else if (j == 9) {
      map[i][j].name = "GO TO JAIL";
      map[i][j].type = "Government";
      map[i][j].value = 0;
      map[i][j].owner = 0;


    }
  } else {
    if (i == 1) {
      if (j == 0) {
        map[i][j].name = "CHARLES PLACE";
        map[i][j].type = "Purple";
        map[i][j].value = 140;
        map[i][j].owner = 0;
      } else if (j == 9) {
        map[i][j].name = "BOARDWALK";
        map[i][j].type = "Blue";
        map[i][j].value = 400;
        map[i][j].owner = 0;
      } else {
        map[i][j].name = "";
        map[i][j].type = "";
        map[i][j].value = 0;
        map[i][j].owner = 0;
```

```
   }
 } else if (i == 2) {
   if (j == 0) {
     map[i][j].name = "STATES AVE";
     map[i][j].type = "Purple";
     map[i][j].value = 140;
     map[i][j].owner = 0;
   } else if (j == 9) {
     map[i][j].name = "PARK PLACE";
     map[i][j].type = "Blue";
     map[i][j].value = 350;
     map[i][j].owner = 0;
   } else {
     map[i][j].name = "";
     map[i][j].type = "";
     map[i][j].value = 0;
     map[i][j].owner = 0;
   }
 } else if (i == 3) {
   if (j == 0) {
     map[i][j].name = "VIRGINIA AVE";
     map[i][j].type = "Purple";
     map[i][j].value = 160;
```

```
      map[i][j].owner = 0;
    } else if (j == 9) {
      map[i][j].name = "CHANCE";
      map[i][j].type = "Coin";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    } else {
      map[i][j].name = "";
      map[i][j].type = "";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    }
  } else if (i == 4) {
    if (j == 0) {
      map[i][j].name = "PA RAILROAD";
      map[i][j].type = "Railroad";
      map[i][j].value = 200;
      map[i][j].owner = 0;
    } else if (j == 9) {
      map[i][j].name = "LINE RAILROAD";
      map[i][j].type = "Railroad";
      map[i][j].value = 200;
      map[i][j].owner = 0;
    } else {
      map[i][j].name = "";
      map[i][j].type = "";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    }
  } else if (i == 5) {
    if (j == 0) {
      map[i][j].name = "JAMES PLACE";
      map[i][j].type = "Orange";
      map[i][j].value = 180;
      map[i][j].owner = 0;
    } else if (j == 9) {
      map[i][j].name = "PA AVE";
      map[i][j].type = "Green";
      map[i][j].value = 320;
      map[i][j].owner = 0;
    } else {
      map[i][j].name = "";
      map[i][j].type = "";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    }
```

```
} else if (i == 6) {
  if (j == 0) {
    map[i][j].name = "COMM. CHEST";
    map[i][j].type = "Chest";
    map[i][j].value = 0;
    map[i][j].owner = 0;
  } else if (j == 9) {
    map[i][j].name = "COMM. CHEST";
    map[i][j].type = "Chest";
    map[i][j].value = 0;
    map[i][j].owner = 0;
  } else {
    map[i][j].name = "";
    map[i][j].type = "";
    map[i][j].value = 0;
    map[i][j].owner = 0;
  }
} else if (i == 7) {
  if (j == 0) {
    map[i][j].name = "TN AVE";
    map[i][j].type = "Orange";
    map[i][j].value = 180;
    map[i][j].owner = 0;
    } else if (j == 9) {
      map[i][j].name = "NC AVE";
      map[i][j].type = "Green";
      map[i][j].value = 300;
      map[i][j].owner = 0;
    } else {
      map[i][j].name = "";
      map[i][j].type = "";
      map[i][j].value = 0;
      map[i][j].owner = 0;
    }
  } else {
    if (j == 0) {
      map[i][j].name = "NY AVE";
      map[i][j].type = "Orange";
      map[i][j].value = 200;
      map[i][j].owner = 0;
    } else if (j == 9) {
      map[i][j].name = "PACIFIC AVE";
      map[i][j].type = "Green";
      map[i][j].value = 300;
      map[i][j].owner = 0;
    } else {
          map[i][j].name = "";

          map[i][j].type = "";

          map[i][j].value = 0;

          map[i][j].owner = 0;

        }

      }

    }

  }


 }


 return map;

}


void printMap(Property ** & map, Player * players) { //This function prints the entire structure array
(map) after changes

 int len;

 bool pos;

 string leng;

 for (int i = 0; i < 10; i++) { //Printing the top of the cubes at the top of the map

  cout << "|--------------|";

 }

 cout << endl;

 for (int i = 0; i < 10; i++) {
  len = map[9][i].name.length(); //Getting the length of the map name for formatting

  if (len % 2 == 0) {



    cout << fixed << "|" << setw((14 - len) / 2 + len) << map[9][i].name << setw((14 - len) / 2 + 1)
<< "|"; //Printing the map name in the cube

  } else {

    cout << fixed << "|" << setw((14 - len) / 2 + len) << map[9][i].name << setw((14 - len) / 2 + 2)
<< "|";//Printing the map name in the cube

  }

 }

 cout << endl;



 for (int i = 0; i < 10; i++) {

  leng = "$" + to_string(map[9][i].value); //Creating a string of $ + the value of the map

  len = leng.length(); //Getting the length of that string

  if (map[9][i].type == "Government" || map[9][i].type == "Chest" || map[9][i].type == "Coin")
{ //Checking if the map does not have a value

    cout << "|              |";

  } else {

    if (len % 2 == 0) {



      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
//Printing the value of the map in the cube

    } else {

      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
//Printing the value of the map in the cube
```

```
      }

    }

  }

  cout << endl;

  pos = false; //Used to check if the player is on this cube of the map

  if (players[0].token.position[0] == players[1].token.position[0] && players[0].token.position[1] ==
players[1].token.position[1]) { //Checking whether both players are on the same part of the map

    for (int i = 0; i < 10; i++) {

      leng = players[0].token.name + ("(P1)"); //Creating a string of the token name + P1

      len = leng.length(); //length of that string

      if (map[players[0].token.position[0]][players[0].token.position[1]].name == map[9][i].name)
{ //Checking whether the player is on this cube



        pos = true; //Positive

        if (len % 2 == 0) {



          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
//Printing the player's name

        } else {

          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) <<
"|";//Printing the player's name

        }

      }



      if (pos == false) {
```

```
    cout << "|              |"; //Printing emptiness since player is not on this cube
   }
   pos = false;
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
   leng = players[1].token.name + ("(P2)"); //P2's turn
   len = leng.length();

   if (map[players[1].token.position[0]][players[1].token.position[1]].name == map[9][i].name) {

    pos = true;
    if (len % 2 == 0) {

     cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
    } else {
     cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
    }
   }

   if (pos == false) {
    cout << "|              |";
   }
```

```
    pos = false;
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
    cout << "|                |";
  }

} else {
  for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 2; j++) {
      if (j == 0) {
        leng = players[0].token.name + ("(P1)");
        len = leng.length();
      } else {
        leng = players[1].token.name + ("(P2)");
        len = leng.length();
      }

      if (map[players[j].token.position[0]][players[j].token.position[1]].name == map[9][i].name) {

        pos = true;
        if (len % 2 == 0) {
          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
        } else {
          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
        }
      }

    }
    if (pos == false) {
      cout << "|              |";
    }
    pos = false;
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
    cout << "|              |";
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
    cout << "|              |";
  }
}
cout << endl;
for (int i = 0; i < 10; i++) {
    leng = "Set:" + map[9][i].type; //Printing the set
    len = leng.length();
    if (map[9][i].type == "Government" || map[9][i].type == "Chest" || map[9][i].type == "Coin") {
      cout << "|              |";
    } else {
      if (len % 2 == 0) {

        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
      } else {
        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
      }
    }
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
    cout << "|--------------|";
  }

  cout << endl;
  for (int i = 0; i < 10; i++) {
    if (map[9][i].owner == 1 || map[9][i].owner == 2) leng = "Owner: P" +
to_string(map[9][i].owner); //Printing the owner
    else leng = "Owner: None";
    len = leng.length();
  if (map[9][i].type == "Government" || map[9][i].type == "Chest" || map[9][i].type == "Coin") {
    cout << "|              |";
  } else {
    if (len % 2 == 0) {

      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
    } else {
      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
    }
  }
}
cout << endl;
for (int i = 0; i < 10; i++) {
  cout << "|--------------|";
}
cout << endl;

for (int i = 8; i > 0; i--) {
  for (int j = 0; j < 10; j++) {
    len = map[i][j].name.length();
    if (j == 0) {
      cout << fixed << "|--------------|" << setw(16 * 9) << "|--------------|" << endl;
      if (len % 2 == 0) {
          cout << fixed << "|" << setw((14 - len) / 2 + len) << map[i][j].name << setw((14 - len) / 2 + 1)
<< "|";

        } else {

          cout << fixed << "|" << setw((14 - len) / 2 + len) << map[i][j].name << setw((14 - len) / 2 + 2)
<< "|";

        }



      } else if (j == 9) {

        if (map[i][j].name.length() % 2 == 0) {



          cout << fixed << "|" << setw((14 - len) / 2 + map[i][j].name.length()) << map[i][j].name <<
setw((14 - len) / 2 + 1) << "|";

        } else {

          cout << fixed << "|" << setw((14 - len) / 2 + map[i][j].name.length()) << map[i][j].name <<
setw((14 - len) / 2 + 2) << "|";

        }

        cout << endl;



        leng = "$" + to_string(map[i][0].value);

        len = leng.length();

        if (map[i][0].type == "Government" || map[i][0].type == "Chest" || map[i][0].type == "Coin") {

          cout << "|              |";

        } else {

          if (len % 2 == 0) {
```

```
          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";

        } else {

          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";

        }

      }

      leng = "$" + to_string(map[i][j].value);

      len = leng.length();


      if (map[i][j].type == "Government" || map[i][j].type == "Chest" || map[i][j].type == "Coin") {

        cout << setw(16 * 9) << "|              |";

      } else {

        if (len % 2 == 0) {


          cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 1) << "|";

        } else {

          cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 2) << "|";

        }

      }

      cout << endl;


      if (players[0].token.position[0] == players[1].token.position[0] && players[0].token.position[1]
== players[1].token.position[1]) {
        leng = players[0].token.name + ("(P1)");

        len = leng.length();

        if (map[players[0].token.position[0]][players[0].token.position[1]].name == map[i][0].name) {



          if (len % 2 == 0) {



            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|"
<< setw(16 * 9) << "|              |" << endl;

            leng = players[1].token.name + ("(P2)");

            len = leng.length();

            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|"
<< setw(16 * 9) << "|              |" << endl;

          } else {

            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|"
<< setw(16 * 9) << "|              |" << endl;

            leng = players[1].token.name + ("(P2)");

            len = leng.length();

            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|"
<< setw(16 * 9) << "|              |" << endl;

          }

        } else if (map[players[0].token.position[0]][players[0].token.position[1]].name ==
map[i][9].name) {

          if (len % 2 == 0) {
            cout << fixed << "|              |" << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) <<
leng << setw((14 - len) / 2 + 1) << "|" << endl;

            leng = players[1].token.name + ("(P2)");

            len = leng.length();

            cout << fixed << "|              |" << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) <<
leng << setw((14 - len) / 2 + 1) << "|" << endl;

          } else {

            cout << fixed << "|              |" << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) <<
leng << setw((14 - len) / 2 + 2) << "|" << endl;

            leng = players[1].token.name + ("(P2)");

            len = leng.length();

            cout << fixed << "|              |" << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) <<
leng << setw((14 - len) / 2 + 2) << "|" << endl;

          }

        } else {

          cout << "|              |" << setw(16 * 9) << "|              |" << endl;

          cout << "|              |" << setw(16 * 9) << "|              |" << endl;

        }


      } else {


        if (map[players[0].token.position[0]][players[0].token.position[1]].name == map[i][0].name) {

          leng = players[0].token.name + ("(P1)");

          len = leng.length();

          if (len % 2 == 0) {
```

```
            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";

          } else {

            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";

          }

        } else if (map[players[1].token.position[0]][players[1].token.position[1]].name ==
map[i][0].name) {

          leng = players[1].token.name + ("(P2)");

          len = leng.length();



          if (len % 2 == 0) {



            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";

          } else {

            cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";

          }

        } else {

          cout << "|              |";

        }

        if (map[players[0].token.position[0]][players[0].token.position[1]].name == map[i][j].name) {

          leng = players[0].token.name + ("(P1)");

          len = leng.length();



          if (len % 2 == 0) {
```

```
        cout << fixed << setw(16 * 8) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len)
/ 2 + 1) << "|" << endl;

        cout << fixed << "|              |" << setw(16 * 9) << "|              |" << endl;

       } else {

        cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 2) << "|" << endl;

        cout << fixed << "|              |" << setw(16 * 9) << "|              |" << endl;

       }

      } else if (map[players[1].token.position[0]][players[1].token.position[1]].name ==
map[i][j].name) {

       leng = players[1].token.name + ("(P2)");

       len = leng.length();

       if (len % 2 == 0) {



        cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 1) << "|" << endl;

        cout << fixed << "|              |" << setw(16 * 9) << "|              |" << endl;

       } else {

        cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 2) << "|" << endl;

        cout << fixed << "|              |" << setw(16 * 9) << "|              |" << endl;

       }

      } else {

       cout << fixed << setw(16 * 9) << "|              |" << endl;

       cout << fixed << "|              |" << setw(16 * 9) << "|              |" << endl;
```

```
  }


}
cout << fixed << "|              |" << setw(16 * 9) << "|              |" << endl;


leng = "Set:" + map[i][0].type;
len = leng.length();
if (map[i][0].type == "Government" || map[i][0].type == "Chest" || map[i][0].type == "Coin") {
  cout << "|              |";
} else {
  if (len % 2 == 0) {


    cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
  } else {
    cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
  }
}
leng = "Set:" + map[i][j].type;
len = leng.length();
if (map[i][j].type == "Government" || map[i][j].type == "Chest" || map[i][j].type == "Coin") {
  cout << setw(16 * 9) << "|              |";
} else {
  if (len % 2 == 0) {
      cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 1) << "|";

     } else {

      cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 2) << "|";

     }

    }

    cout << endl;


    cout << fixed << "|-------------|" << setw(16 * 9) << "|-------------|" << endl;

    //Owner

    if (map[i][0].owner == 0) leng = "Owner: None";

    else leng = "Owner: P" + to_string(map[i][0].owner);


    len = leng.length();

    if (map[i][0].type == "Government" || map[i][0].type == "Chest" || map[i][0].type == "Coin") {

     cout << "|             |";

    } else {

     if (len % 2 == 0) {


      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";

     } else {

      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
```

```
          }
        }
        if (map[i][j].owner == 0) leng = "Owner: None";
        else leng = "Owner: P" + to_string(map[i][j].owner);


        len = leng.length();
        if (map[i][j].type == "Government" || map[i][j].type == "Chest" || map[i][j].type == "Coin") {
          cout << setw(16 * 9) << "|              |";
        } else {
          if (len % 2 == 0) {


            cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 1) << "|";
          } else {
            cout << fixed << setw(16 * 8 + 1) << "|" << setw((14 - len) / 2 + len) << leng << setw((14 -
len) / 2 + 2) << "|";
          }
        }
        cout << endl;
        cout << fixed << "|--------------|" << setw(16 * 9) << "|--------------|" << endl;


      } else {
        cout << fixed << setw(16) << "";
      }
```

```
    }
  }


  for (int i = 0; i < 10; i++) {
    cout << "|--------------|";
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
    len = map[0][i].name.length();
    if (len % 2 == 0) {


      cout << fixed << "|" << setw((14 - len) / 2 + len) << map[0][i].name << setw((14 - len) / 2 + 1)
<< "|";
    } else {
      cout << fixed << "|" << setw((14 - len) / 2 + len) << map[0][i].name << setw((14 - len) / 2 + 2)
<< "|";
    }
  }
  cout << endl;
  for (int i = 0; i < 10; i++) {
    leng = "$" + to_string(map[0][i].value);
    len = leng.length();
    if (map[0][i].type == "Government" || map[0][i].type == "Chest" || map[0][i].type == "Coin") {
      cout << "|              |";
    } else {
      if (len % 2 == 0) {


        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
      } else {
        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
      }
    }
  }
  cout << endl;
  pos = false;
  if (players[0].token.position[0] == players[1].token.position[0] && players[0].token.position[1] ==
players[1].token.position[1]) {
    for (int i = 0; i < 10; i++) {
      leng = players[0].token.name + ("(P1)");
      len = leng.length();
      if (map[players[0].token.position[0]][players[0].token.position[1]].name == map[0][i].name) {


        pos = true;
        if (len % 2 == 0) {


          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
        } else {
          cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
    }
  }

  if (pos == false) {
    cout << "|              |";
  }
  pos = false;
}
cout << endl;
for (int i = 0; i < 10; i++) {
  leng = players[1].token.name + ("(P2)");
  len = leng.length();

  if (map[players[1].token.position[0]][players[1].token.position[1]].name == map[0][i].name) {

    pos = true;
    if (len % 2 == 0) {

      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
    } else {
      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
    }
  }
```

```
      if (pos == false) {
        cout << "|               |";
      }
      pos = false;
    }
    cout << endl;
    for (int i = 0; i < 10; i++) {
      cout << "|               |";
    }
  } else {
    for (int i = 0; i < 10; i++) {
      for (int j = 0; j < 2; j++) {
        if (j == 0) {
          leng = players[0].token.name + ("(P1)");
          len = leng.length();
        } else {
          leng = players[1].token.name + ("(P2)");
          len = leng.length();
        }

        if (map[players[j].token.position[0]][players[j].token.position[1]].name == map[0][i].name) {
```

```
      pos = true;
      if (len % 2 == 0) {


        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";
      } else {
        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";
      }
    }


  }
  if (pos == false) {
    cout << "|              |";
  }
  pos = false;
}
cout << endl;
for (int i = 0; i < 10; i++) {
  cout << "|              |";
}
cout << endl;
for (int i = 0; i < 10; i++) {
  cout << "|              |";
}
```

```
}

cout << endl;

for (int i = 0; i < 10; i++) { //Bot

  leng = "Set:" + map[0][i].type;

  len = leng.length();

  if (map[0][i].type == "Government" || map[0][i].type == "Chest" || map[0][i].type == "Coin") {

    cout << "|              |";

  } else {

    if (len % 2 == 0) {


      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";

    } else {

      cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";

    }

  }

}


cout << endl;

for (int i = 0; i < 10; i++) {

  cout << "|--------------|";

}

cout << endl;

for (int i = 0; i < 10; i++) {
    if (map[0][i].owner == 1 || map[0][i].owner == 2) leng = "Owner: P" +
to_string(map[0][i].owner);

    else leng = "Owner: None";

    len = leng.length();

    if (map[0][i].type == "Government" || map[0][i].type == "Chest" || map[0][i].type == "Coin") {

      cout << "|             |";

    } else {

      if (len % 2 == 0) {


        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 1) << "|";

      } else {

        cout << fixed << "|" << setw((14 - len) / 2 + len) << leng << setw((14 - len) / 2 + 2) << "|";

      }

    }

  }

  cout << endl;

  for (int i = 0; i < 10; i++) {

    cout << "|-------------|";

  }

  cout << endl;

}


void chanceCoin(Player * players, int i) {

  int chance = 0;
  string answer;

  cout << "You landed on a chance coin. Are you ready to pull your card? Enter anything to
continue." << endl;

  getline(cin, answer);

  chance = rand() % 3;


  switch (chance) {

  case 0:

    cout << "Advance to GO. You gain $200." << endl;

    players[i].token.position[0] = 0;

    players[i].token.position[1] = 9;

    players[i].money = players[i].money + 200;

    cout << "Your new balance is $" << players[i].money << endl;

    break;


  case 1:

    cout << "Go directly to jail. You will not gain the $200 from GO." << endl;

    players[i].token.position[0] = 9;

    players[i].token.position[1] = 0;

    players[i].jail = true;

    break;


  case 2:

    cout << "You have been elected mayor. Give $50 to the other player." << endl;
    if (i == 0) {

      players[i].money = players[i].money - 50;

      players[i + 1].money = players[i + 1].money + 50;

    } else {

      players[i].money = players[i].money - 50;

      players[i - 1].money = players[i - 1].money + 50;

    }

    cout << "Your new balance is $" << players[i].money << endl;

    break;


  }

}


void communityChest(Player * players, int i) {

  int chest = 0;

  string answer;

  cout << "You landed on a community chest. Are you ready to pull your card? Enter anything to
continue." << endl;

  getline(cin, answer);

  chest = rand() % 3;


  switch (chest) {

  case 0:

    cout << "Pay a hospital bill of $100." << endl;
    players[i].money = players[i].money - 100;

    cout << "Your new balance is $" << players[i].money << endl;

    break;


  case 1:

    cout << "You inherit $200 from a lost grandparent!" << endl;

    players[i].money += 200;

    cout << "Your new balance is $" << players[i].money << endl;

    break;


  case 2:

    cout << "You hacked into the other player's bank. You stole $50." << endl;

    if (i == 0) {

      players[i].money = players[i].money + 50;

      players[i + 1].money = players[i + 1].money - 50;

    } else {

      players[i].money = players[i].money + 50;

      players[i - 1].money = players[i - 1].money + 50;

    }

    cout << "Your new balance is $" << players[i].money << endl;

    break;


  }
```

```
}


bool gameOver(Property ** map, Player * players, int i, bool game, fstream & file) { //Checks
whether game is over

 int ownercount = 0; //Used to count whether an entire row is owned by a player (win)

 int ownercount2 = 0; //Used to count whether an entire row is owned by a player (win)

 int railroadcount = 0; //Used to count whether all railroads are owned by a player (win)



 string winner;

 winner = players[i].token.name;

 file.open("leaderboard.txt", ios::out | ios::in | ios::binary);

 players[i].wins.open("wins.txt", ios::out);



 for (int g = 0; g < 9; g++) {

  if (g == 0) {

   ownercount = 0;

   railroadcount = 0;

   for (int l = 0; l < 10; l++) {

    if (map[g][l].type == "Government" || map[g][l].type == "Chest" || map[g][l].type == "Coin") {



    } else if (map[g][l].type == "Railroad") {

     if (map[g][l].owner == i + 1) {

      railroadcount++;

     }
```

```
      } else {
        if (map[g][l].owner == i + 1) {
          ownercount++;
        }
      }
    }
    if (ownercount == 4) {
      winner = players[i].token.name;
      players[i].wins << "+1";
      file.seekg(0L, ios::beg);
      file.write(reinterpret_cast < char * > ( & winner), sizeof( & winner));
      file.read(reinterpret_cast < char * > ( & winner), sizeof( & winner));

      cout << winner << " has won!" << endl;

      game = false;
    }

  } else if (g == 9) {
    ownercount = 0;

    for (int l = 0; l < 10; l++) {
      if (map[g][l].type == "Government" || map[g][l].type == "Chest" || map[g][l].type == "Coin") {
      } else if (map[g][l].type == "Railroad") {
        if (map[g][l].owner == i + 1) {
          railroadcount++;
        }
      } else {
        if (map[g][l].owner == i + 1) {
          ownercount++;
        }
      }
    }
    if (ownercount == 5) {
      winner = players[i].token.name;
      players[i].wins << "+1";
      file.seekg(0L, ios::beg);
      file.write(reinterpret_cast < char * > ( & winner), sizeof( & winner));
      file.read(reinterpret_cast < char * > ( & winner), sizeof( & winner));

      cout << winner << " has won!" << endl;
      game = false;
    }

  } else {
```

```
ownercount = 0;

ownercount2 = 0;

for (int l = 0; l < 10; l = l + 9) {

 if (l == 0) {

  if (map[g][l].type == "Government" || map[g][l].type == "Chest" || map[g][l].type == "Coin") {


  } else if (map[g][l].type == "Railroad") {

   if (map[g][l].owner == i + 1) {

    railroadcount++;

   }

  } else {

   if (map[g][l].owner == i + 1) {

    ownercount++;

   }

  }

 }

 if (l == 9) {

  if (map[g][l].type == "Government" || map[g][l].type == "Chest" || map[g][l].type == "Coin") {


  } else if (map[g][l].type == "Railroad") {

   if (map[g][l].owner == i + 1) {

    railroadcount++;

   }
```

```
          } else {
            if (map[g][l].owner == i + 1) {
              ownercount2++;
            }
          }
        }
      }
      if (ownercount == 5 || ownercount2 == 4 || railroadcount == 3) {
        winner = players[i].token.name;
        players[i].wins << "+1";
        file.seekg(0L, ios::beg);
        file.write(reinterpret_cast < char * > ( & winner), sizeof( & winner));
        file.read(reinterpret_cast < char * > ( & winner), sizeof( & winner));

        cout << winner << " has won!" << endl;

        game = false;


      }
    }
  }
  if (players[i].money < 0) {
    game = false;
```

```
 }
 file.close();
 players[i].wins.close();
 return game;
}
```