Project 1

<Monopoly>

CIS-17C 42513
Name: Abu-Ghazaleh, Zayd
Date: 5/16/2025

Introduction & Game Rules

Title: Monopoly

For this project, I created monopoly.
Although, this is a project that I am continuing from CIS 17A. I added many functions that utilize the STL library that we were tasked with this time.

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.

Stats

Project size: about 1533 lines
The number of functions: 6

The number of classes: 3.

Approach to Development

This project demonstrates almost every concept that we have learned so far (in 17A and 17C)
This project can also easily implement more complex concepts such as hashes and recursions for the next iteration.
These can be used to cut down on some parts of the code and create some plugins and functions as well.

This project took me about 3 weeks in 17C, and another week and a half now, as this project was a lot more complex than it appears.
I ran into errors that took hefty amounts of time to fix, and simply adding some new functions such as match data proved difficult. This is because monopoly is a complex game that is difficult to implement into c++ with the knowledge I have now.
I'm not completely happy with how the project is now, but I believe I can improve on the code and change that in the next iteration.
I have included different iterations of this project in the folder.
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.
The new concepts that I had to construct include a lot of different things from the STL library.

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.

There are three classes, titled Piece, Player and Property.

The Piece class contains the code

```
class Piece{
public:
   string name; //Name of the piece
    int position[1]; //Position of the piece

};
```

The Player class contains the code

```
class Player{
public:
   Piece token; //The piece the player is moving
    int money; //The amount of money the player has
    bool turn; //Whether it is this player's turn
    bool jail; //Whether this player is in jail
    fstream wins; //A file holding the wins
};
```

The Property class contains the code

```
struct Property{
   string name; //Name of the property
    string type; //Type of property
    int value; //Value of the property
    int owner; //Owner of the property
};
```

They are each in their respective header file, and the code is organized into functions and code for the main game in main.

Sample Input and Output

The code prints out the entire monopoly board on the screen every time a player rolls, so I will just write an example of an input and output.

Output: Asks the players to name their pieces
Input: Players input piece names
Output: Prints out monopoly board with prices and player positions, then asks the player if they are ready to roll
Input: Type anything to continue
Output: Outputs the updated map, the rolls, and then asks player if they want to buy a property for "" price if they landed on a property or asks the player to pull a card/chance coin if they landed on those.
Input: Input 'yes' or 'no' to buy property, input anything to pull a card/chance coin.
Output: Outputs updated map (new ownership) and balance if they bought, then asks player 2 if they are ready to roll.
Input: Type anything to continue
Output: Continues from the 3rd output, until the game is completed.

Checkoff Sheet

1.Container classes (Where in code did you put each of these Concepts and how were they used?

1.Sequences (At least 1)

List:

I used the list sequence to hold the total rolls, double rolls, and distance travelled which is then printed out in the data tab.
The lists were initialized on lines 46, 58, and 59. An example of them being used is from lines 174 to 220.

2.Associative Containers (At least 2)

Set:

I used the set container to hold, update, and sort expenses throughout the game and also to show expenses in the data tab.
The sets were initialized on line 54. An example of them being used is on lines 154, 155, 335, and 345.

map

I used the map container to hold balances for player 1 and player 2, tying the balances to their strings. This was also printed out in the data tab. It was initialized on line 47, and an example of it being used is on lines 103,104, and 118-152.

3.Container adaptors (At least 2)

Stack:

I used the stack container to hold and update the strings for every purchase in the game. They were initialized on line 48, and an example of them being used is from line 157 to line 167.

Queue:

I used the queue to hold prices throughout the game, updating based on recently pinged. An example of them being used is lines 161, 167, 341, and 351.

2.Iterators

1.Concepts (Describe the iterators utilized for each Container)

1.Trivial Iterator

The trivial iterator is just an object that may be dereferenced as a way to see another object. An example in my code is line 181.

2.Input Iterator

The input iterator is an iterator that can only read from its container, unable to output or move in multiple directions. An example in my code is from lines 145-152.

3.Output Iterator

The output iterator is an iterator that can only output to its container. An example in my code is lines 134-138.

4.Forward Iterator

A forward iterator is an iterator that can dereference and increment, but can not move backwards. An example in my code is also lines 134-138.

5.Bidirectional Iterator

A bidirectional Iterator can move both backwards and forwards in a container. An example in my code is on line 197 and what comes after.

6.Random Access Iterator

A random access iterator can move forwards, backwards, output and input, and can access any element of a container. This is used in my code from 196-220.

3.Algorithms (Choose at least 1 from each category)

1.Non-mutating algorithms

Count: This algorithm is used to count the amount of times a number/string appears. I use this in my code to count the amount of identical rolls. This is on line 198 and 206.

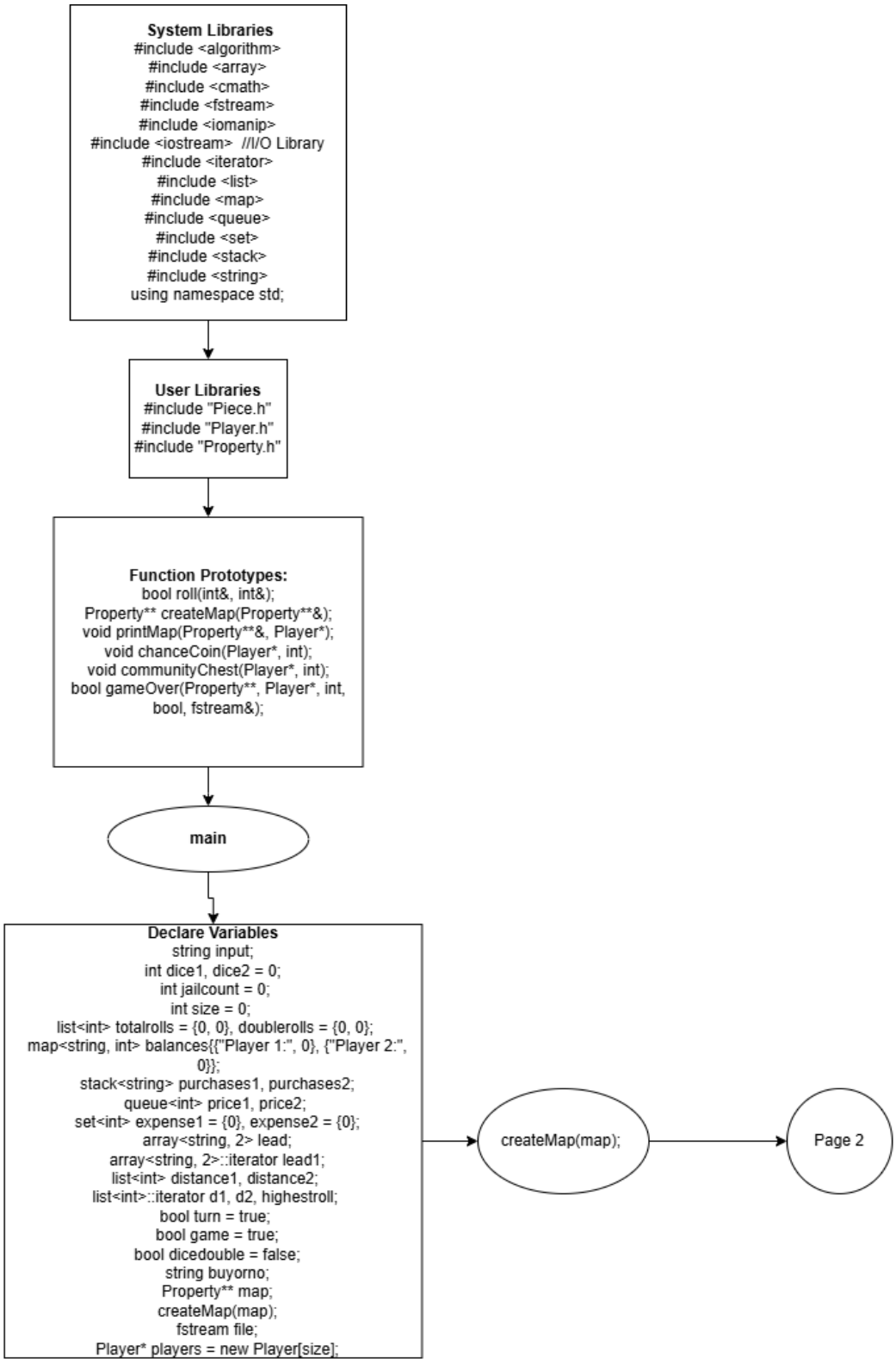
2.Mutating algorithms

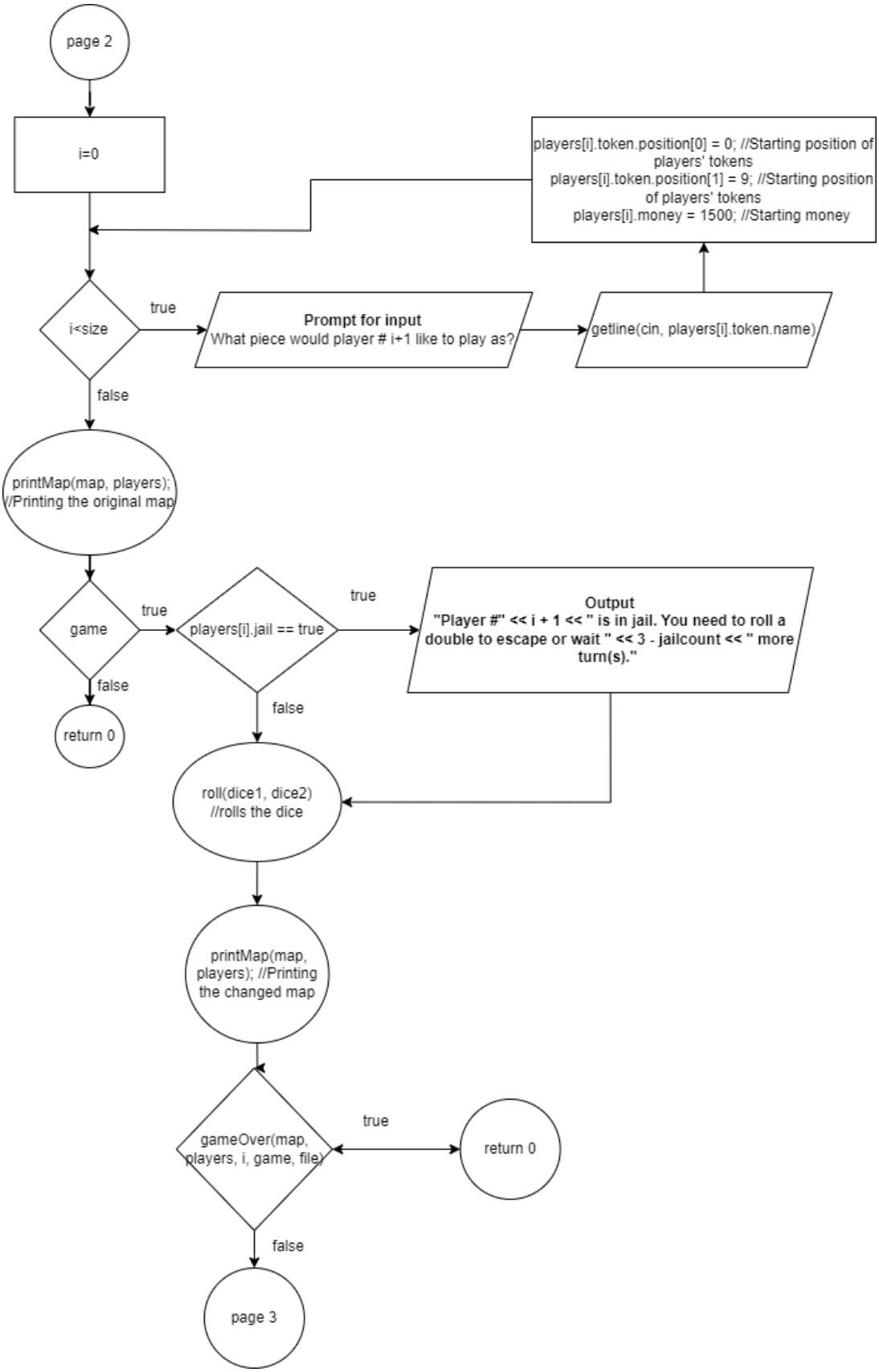
Fill: This algorithm fills a container with a number/string. I use this to fill my arrays. An example is on line 56.

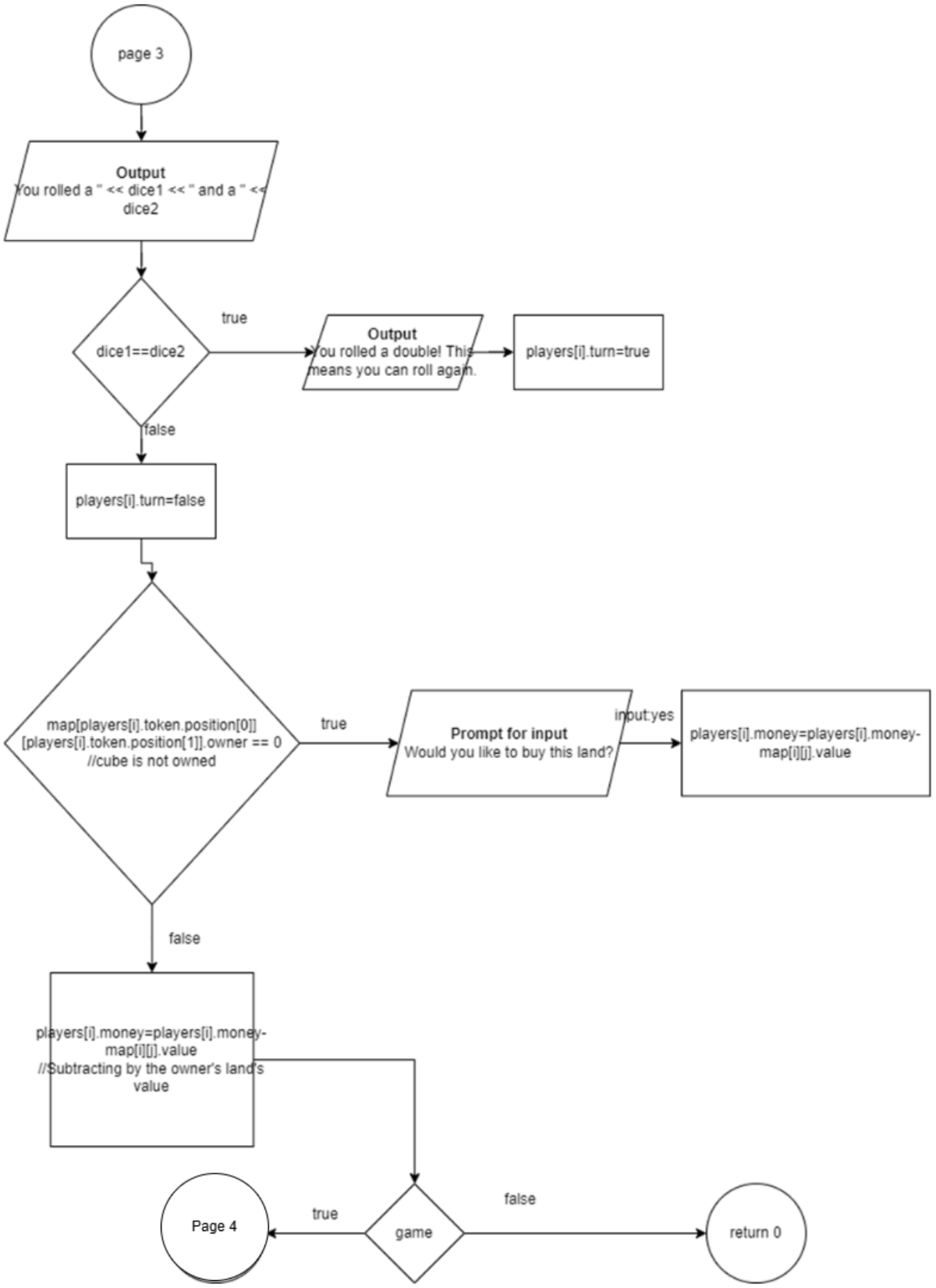
3.Organization

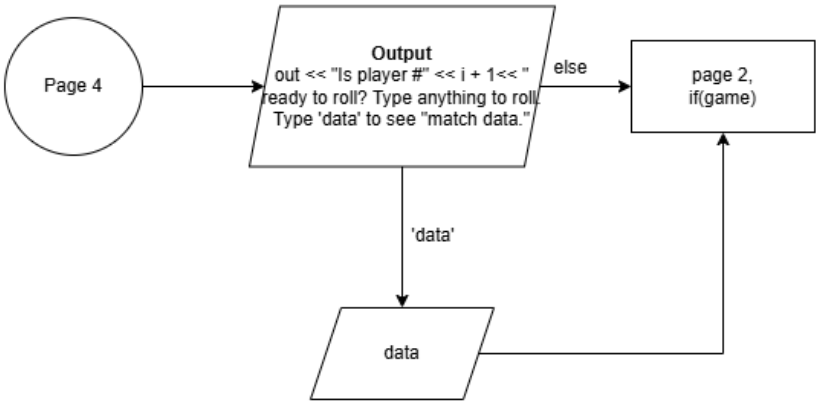
Minimum and maximum: This algorithm finds the max and min of a container. I use this to find the highest roll, on lines 183 and 212.

Flow Chart (Next page)


System Libraries
#include <algorithm>
#include <array>
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream> ///I/O Library
#include <iterator>
#include <list>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
using namespace std;
User Libraries
#include "Piece.h"
#include "Player.h"
#include "Property.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&);
main
Declare Variables
string input;
int dice1, dice2 = 0;
int jailcount = 0;
int size = 0;
list<int> totalrolls = {0, 0}, doublerolls = {0, 0};
map<string, int> balances{{"Player 1:", 0}, {"Player 2:",
0}};
stack<string> purchases1, purchases2;
queue<int> price1, price2;
set<int> expense1 = {0}, expense2 = {0};
array<string, 2> lead;
array<string, 2>::iterator lead1;
list<int> distance1, distance2;
list<int>::iterator d1, d2, highestroll;
bool turn = true;
bool game = true;
bool dicedouble = false;
string buyorno;
Property** map;
createMap(map);
fstream file;
Player* players = new Player[size];
createMap(map);
Page 2


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
Page 4
true
game
false
return 0


Page 4
Output
out << "Is player #" << i + 1<< "
ready to roll? Type anything to roll
Type 'data' to see "match data."
else
page 2,
if(game)
'data'
data

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 if the player wants to see data

Print all data (balances, tiles traveled, property owned, etc.)

Move onto the next player

Else

Return 0;

Reference

1. Textbook
2. Lectures
3. Monopoly Board

Program

```
// System Libraries

#include <algorithm>

#include <array>

#include <cmath>

#include <fstream>

#include <iomanip>

#include <iostream>  //I/O Library

#include <iterator>

#include <list>

#include <map>

#include <queue>
```

```
#include <set>
#include <stack>
#include <string>

using namespace std;

// User Libraries
#include "Piece.h"
#include "Player.h"
#include "Property.h"

enum Players { PLAYER1, PLAYER2 };

// Global Constants Only
// Well known Science, Mathematical and Laboratory Constants
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&);
// 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
  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
  list<int> totalrolls = {0, 0}, doublerolls = {0, 0}; //lists used for rolls in data menu
  map<string, int> balances{{"Player 1:", 0}, {"Player 2:", 0}}; //map used for balances in data menu
  stack<string> purchases1, purchases2; //stacks used for purchases in data menu
  purchases1.push("No Properties Purchased");
  purchases2.push("No Properties Purchased");
  queue<int> price1, price2; //queues used for prices in data menu
  price1.push(0);
  price2.push(0);
  set<int> expense1 = {0}, expense2 = {0}; //sets used for expenses in data menu
  array<string, 2> lead; //array used for lead in data menu
  fill(lead.begin(), lead.end(), "Tied");
  array<string, 2>::iterator lead1;//iterator to go through the array
  list<int> distance1, distance2; //lists used for distances in data menu
```

```
list<int>::iterator d1, d2, highestroll; //iterators to go through the lists
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
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
size=2;
cout<<"This game is designed for 2 players."<<endl;
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 << " more turn(s)." << endl;
      }
      balances["Player 1:"] = players[0].money;
      balances["Player 2:"] = players[1].money;
```

```
cout << "Is player #" << i + 1
    << " ready to roll? Type anything to roll. Type 'data' to see "
       "match data."
    << endl;
getline(cin, input);


if (input == "data") {
  cout << "Total Rolls:" << endl
       << "Player 1:" << totalrolls.front() << endl
       << "Player 2:" << totalrolls.back() << endl;
  cout << "Doubles Rolled:" << endl
       << "Player 1:" << doublerolls.front() << endl
       << "Player 2:" << doublerolls.back() << endl;
  cout << "Balances:" << endl;
  std::map<string, int>::iterator it;
  for (it = balances.begin(); it != balances.end(); ++it) {
    cout << it->first << " " << it->second << endl;
  }
  cout << "Ahead (Based on Money):" << endl;
  if (balances["Player 1:"] > balances["Player 2:"]) {
    for (lead1 = lead.begin(); lead1 != lead.end(); ++lead1) {
      if (lead1 == lead.begin()) {
        *lead1 = "Ahead";
```

```
    } else {
      *lead1 = "Behind";
    }
  }
} else if (balances["Player 1:"] < balances["Player 2:"]) {
  for (lead1 = lead.begin(); lead1 != lead.end(); ++lead1) {
    if (lead1 != lead.begin()) {
      *lead1 = "Ahead";
    } else {
      *lead1 = "Behind";
    }
  }
} else {
  for (lead1 = lead.begin(); lead1 != lead.end(); ++lead1) {
    *lead1 = "Tied";
  }
}
for (lead1 = lead.begin(); lead1 != lead.end(); ++lead1) {
  if (lead1 == lead.begin()) {
    cout << "Player 1 is ";
  } else {
    cout << "Player 2 is ";
  }
```

```
  cout << (*lead1) << endl;
}
cout << "Most Expensive Purchases:" << endl;
cout << "Player 1:" << *prev(expense1.end()) << endl;
cout << "Player 2:" << *prev(expense2.end()) << endl;
cout << "Latest Purchases:" << endl;
if (purchases1.top() == "No Properties Purchased") {
  cout << "Player 1:" << purchases1.top() << endl;
} else {
  cout << "Player 1:" << purchases1.top()
       << " at a price of: " << price1.back() << endl;
}
if (purchases2.top() == "No Properties Purchased") {
  cout << "Player 2:" << purchases2.top() << endl;
} else {
  cout << "Player 2:" << purchases2.top()
       << " at a price of: " << price2.back() << endl;
}
cout << "Tiles Traveled In Order By Rolls:" << endl;
cout << "Player 1:";
int freq = 0;
int holder = 0;
```

```
if (!distance1.empty()) {
  for (d1 = distance1.begin(); d1 != distance1.end(); d1++) {
    holder = count(distance1.begin(), distance1.end(), (*d1));
    if (holder > freq) {
      freq = holder;
    }
    if (d1 == prev(distance1.end())) {
      cout << (*d1) << endl;
      cout << "The highest roll was ";
      highestroll = max_element(distance1.begin(), distance1.end());
      cout << (*highestroll) << "." << endl;
      cout << "There were " << freq << " identical rolls!" << endl;
    } else {
      cout << (*d1) << ", ";
    }
  }
} else {
  cout << " None" << endl;
}
freq = 0;
holder = 0;
cout << "Player 2:";
if (!distance2.empty()) {
```

```
  for (d2 = distance2.end(); d2 != distance2.begin(); --d2) {
    holder = count(distance2.begin(), distance2.end(), (*d2));
    if (holder > freq) {
      freq = holder;
    }
    if (d2 != distance2.end()) {
      cout << (*d2) << ", ";
    }
  }
  holder = count(distance2.begin(), distance2.end(), (*d2));
  if (holder > freq) {
    freq = holder;
  }
  cout << (*d2) << endl;
  cout << "The highest roll was ";
  highestroll = max_element(distance2.begin(), distance2.end());
  cout << (*highestroll) << "." << endl;
  cout << "There were " << freq << " identical rolls!" << endl;
 } else {
  cout << " None" << endl;
 }
 cout << endl << "Type anything to continue." << endl;
 getline(cin, input);
```

```
}


if (roll(dice1, dice2) == true) {  // If the roll function returns true,
                                   // player rolled a double
  dicedouble = true;
  if (i == 0) {
    doublerolls.front() = doublerolls.front() + 1;
  } else {
    doublerolls.back() = doublerolls.back() + 1;
  }
} else {
  dicedouble = false;
}
if (players[i].jail ==
    true) {  // Goes through conditions of escaping jail
  if (dicedouble == true) {
    players[i].jail = false;
  } else if (jailcount >= 2) {
    players[i].jail = false;
  } else {
    jailcount++;
  }
}
```

```
if (i == 0) {
  totalrolls.front() = totalrolls.front() + 1;
  distance1.push_back(dice1 + dice2);

} else {
  totalrolls.back() = totalrolls.back() + 1;
  distance2.push_front(dice1 + dice2);
}
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] - (dice1 + dice2) < 0) {
      players[i].token.position[0] =
          (dice1 + dice2) - players[i].token.position[1];
      players[i].token.position[1] = 0;
    } else {
      players[i].token.position[1] =
          players[i].token.position[1] - (dice1 + dice2);
    }
```

```
} else if (players[i].token.position[0] == 9) {
  if (players[i].token.position[1] + (dice1 + dice2) > 9) {
    players[i].token.position[0] =
        9 - ((dice1 + dice2) - (9 - players[i].token.position[1]));
    players[i].token.position[1] = 9;
  } else {
    players[i].token.position[1] =
        players[i].token.position[1] + (dice1 + dice2);
  }
} else {
  if (players[i].token.position[1] == 9) {
    if (players[i].token.position[0] - (dice1 + dice2) < 0) {
      players[i].token.position[1] =
          9 - ((dice1 + dice2) - players[i].token.position[0]);
      players[i].token.position[0] = 0;
    } else {
      players[i].token.position[0] =
          players[i].token.position[0] - (dice1 + dice2);
    }
  } else {
    if (players[i].token.position[0] + (dice1 + dice2) > 9) {
      players[i].token.position[1] =
          (dice1 + dice2) - (9 - players[i].token.position[0]);
```

```
        players[i].token.position[0] = 9;
      } else {
        players[i].token.position[0] =
            players[i].token.position[0] + (dice1 + dice2);
      }
    }
  }
}
printMap(map, players);  // Prints map again
game = gameOver(map, players, i, game,
                file);  // Checks whether game is over
if (game) {
  cout << "You rolled a " << dice1 << " and a " << dice2 << 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 {
    if (i == 0) {
```

```
  expense1.insert(map[players[i].token.position[0]]
                  [players[i].token.position[1]]
                     .value);
  purchases1.push(map[players[i].token.position[0]]
                  [players[i].token.position[1]]
                     .name);
  price1.push(map[players[i].token.position[0]]
                [players[i].token.position[1]]
                  .value);
} else {
  expense2.insert(map[players[i].token.position[0]]
                  [players[i].token.position[1]]
                     .value);
  purchases2.push(map[players[i].token.position[0]]
                  [players[i].token.position[1]]
                     .name);
  price2.push(map[players[i].token.position[0]]
                [players[i].token.position[1]]
                  .value);
}
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;
```

```
        }
      }
    }
  }
  delete[] players;

  for (int i = 0; i < 10; i++) {
    delete[] map[i];
  }
  delete[] map;
  return 0;
}

// Function Implementations

bool roll(int& dice1, int& dice2) {
  bool turn;
  dice1 = rand() % 4 + 1;
  dice2 = rand() % 4 + 1;
  if (dice1 == dice2)
    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] = 0;
      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;
}
```