Project 2

Tittle

Monopoly: Empire Ver 1.2

Course

CSC-17C

Section

42526

Due Date

June 7, 2017

Author

Sili Guo

I. Introduction

Monopoly is a common series of American board game which allow multiple players to roll dices and make movements on the game board. Depending on the luck of player, player can buy land, pay rent, or even get chance card to react differently by following the instruction.

Monopoly: Empire is a game that players collect brand billboards instead of land. Each player has 1000K at the beginning, and tries to use the money properly to get 800 credits of brands. During the time of collecting billboards, players also can attack each other to avoid their opponents to get enough credit.

Also, the chance card in the game plays an important role. Some of the card can help player get more money or attack opponents, but some can be harmful to player himself.

In the game, who collect 800 credits is the winner; however, who run out of money will automatically consider as loser. This asks players to take care of the usage of their money when busy collecting billboards!

II. Game Rules

A. Basic Rules

1.Each player has 1000K at beginning and start at position #0.

2.In each turn, player roll 2 dices, and move corresponding steps.

3.When stop at a position, follow the instruction of that position.

4.[1] represent player, [2] represent computer.

B. Game Board

1.Total 24 positions (0~24) of 7 types.

2.Start

- Every player start from here.
- When pass start point, collect as much money as your total collecting value.

3.Billboard

- Containing name, price, collecting value, and condition(0,1 or 2).
- The first two lines are the name of the brand.
- The third line is price (collecting value), eg.100K (50).
    - Price ≠ collecting value.

    - Price is how much cost you to buy the brand.
    - Collecting value is used for win the game.
- The fourth line is (condition), eg. (1).

  (0 = no owner; 1 = player own; 2 = computer own)
- The fifth line shows player's position, eg.[1] [2].
- The bottom line has the position # of that box.
- Exception: Electric Company Billboard. (See IV. Billboard-E)

4.Chance Card

- Randomly pick a chance card when stop at here.
- Total 8 kinds of chance cards.
- Chance card may not be good for player who pick it.

5.Jail

- To get out of jail, there are two choice.
    - During next turn, pay 100K and get out at once.
    - Roll a double up to 3 turns, if not, pay 50K and get out at 3rd turn.
- Position #19 is go to jail, player will be move to jail on Position #7.

- Simply stop at position #7 is pass by, is not in jail.

6. Take a trip
    - You can spend 100K to move to anywhere on board, or simply do nothing.
    - If you move to a new position, you have to follow the instruction on that position.
7. Tower Tax
    - Tower Tax: return your topmost billboard to board.
    - Rival Tower Tax: return your opponent's topmost billboard to board.

C. Dices

1. One normal dice, one special dice with a 'swap' face.
2. When 'swap' face is up, you have a chance of sneaky swap.

    (sneaky swap: swap your topmost billboard with your opponent's)
3. When you roll a double, you can have another turn until you don't have a double.

    (double: two dices have same number)

D. Billboard

1. 14 billboards total; different price with different collecting value; choose smartly!
2. You can buy billboards when you stop at the corresponding position if the condition is no owner.
3. You have to pay rent fee if you stop at a position whose billboard is owner by your opponents.
4. Nothing will happen if you stop at the position that is owned by your-self.
5. One exception is Electric Company Billboard at position #9.
    - Has 4 in total
    - Doesn't own by anyone.

E. Chance Card

Total 8 cards has different functions.

F. Win

1. Whoever first get 800 collecting value wins.

2.Whoever first run out of money loses.

G. Bonus

The bonus area will be open for player each time when player win a game.

G. Tips For Brand Buying

1. The unit value for different brand is different. Be wise to choose which brand to buy!
2. The brand unit value sorting (See program).

III. Summary

Total Line of Code	**2179**
Utilized from other source	457
Variable	50
Classes	4

Basically, this game is designed for single player playing with computer. There is a player class contains each player's information, including computer, and a game class to control all process happened during the game. I make all the decisions in main function to see the structure clearly. In the new 1.2 version, I utilized two new classes from other source, combined with my original code. One is a hash classes that contains 11 hash functions; the other is a AVL Tree class that helps me insert, balance, and output an AVL Tree.

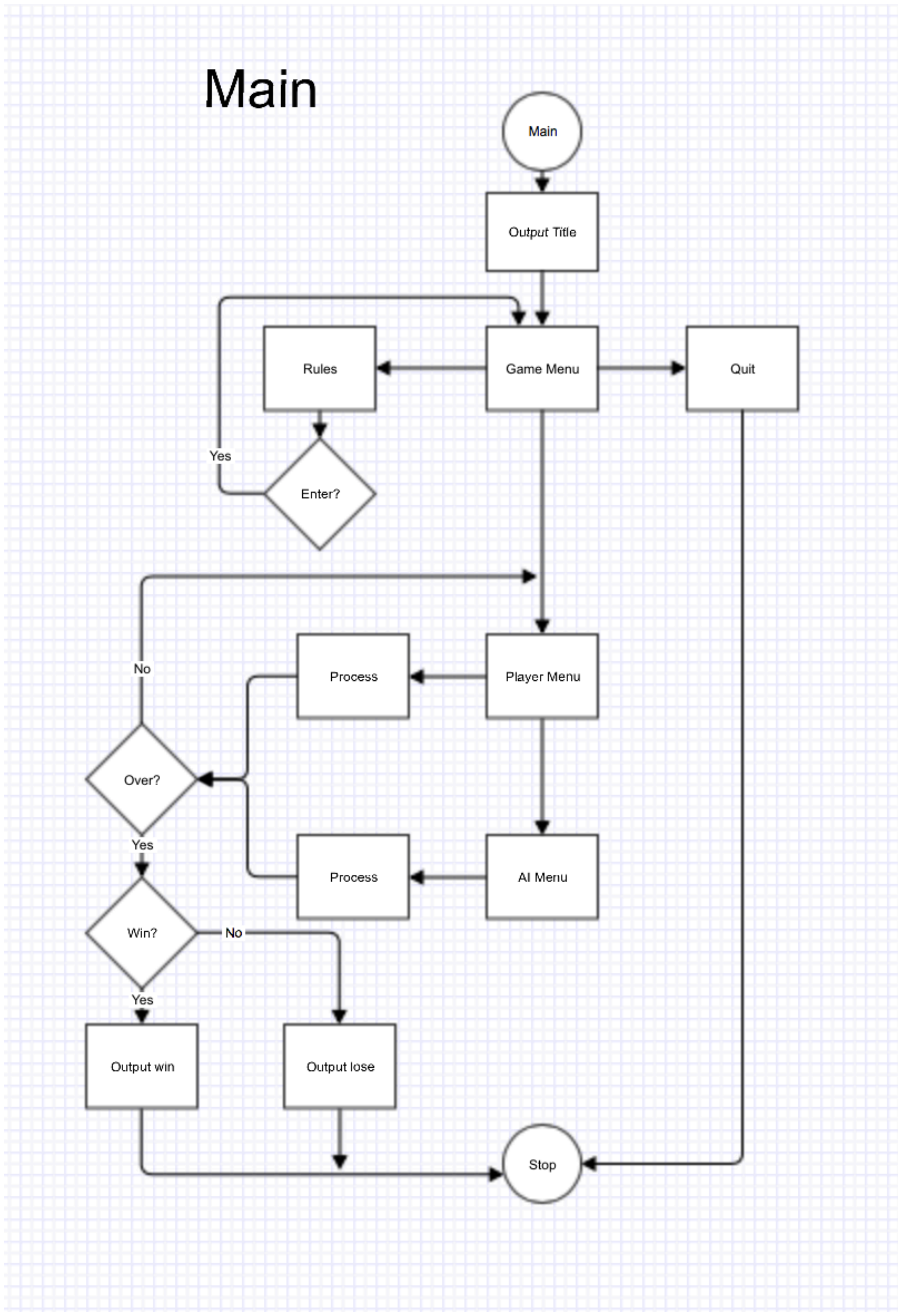
My program basically build on the linked list and its extending usage. In these function, I utilized linked list to contain the collecting value, and use linked list to be act like a stack. I also use map to match the corresponding position #, selling price and collecting value.

In new 1.2 version, I had problems of thinking how to use all the knowledge we learned recently. The first thing I utilized is the hash function. I set a password for player each time start a game, and use merkle tree and hash to test if the password is correct. Then I created a bonus area, and this area will only appear when the player wins the game. In the bonus area, player can see his Billboard values in last game, and even in pre-order, in-order, and post-order of a tree. I also utilized quick sort and map to sort all their brands by their unit value, so that player can see which brand is more worth to buy from “Reading Rules” part. I did not use all methods within Hash and AVL Tree classes, but I kept them so that I can make developments based on them later.

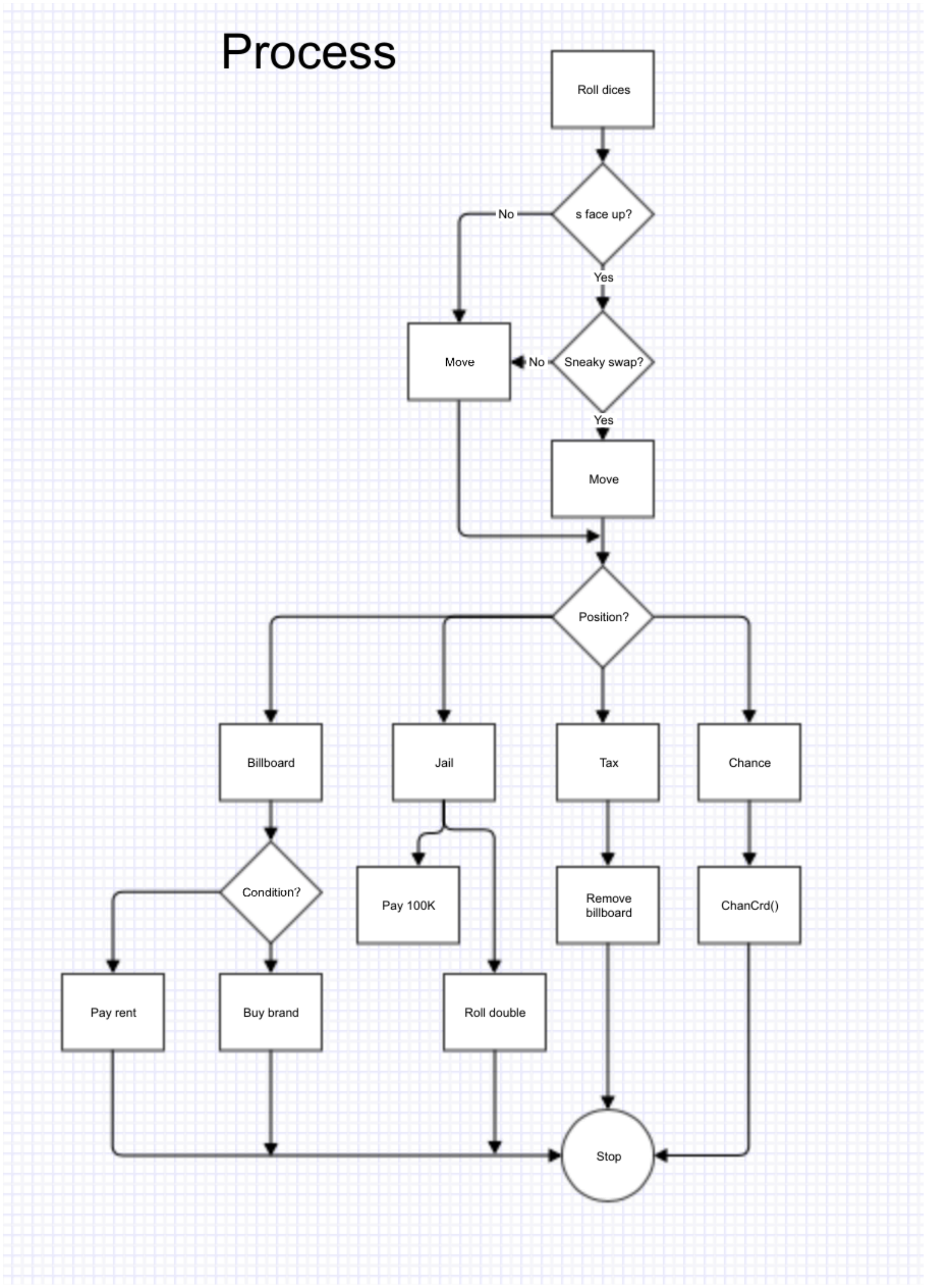
IV. Description

A. Flowcharts

Main:



Process:


Process
Roll dices
s face up?
No
Yes
Move
No
Sneaky swap?
Yes
Move
Position?
Billboard
Jail
Tax
Chance
Condition?
Pay 100K
Remove billboard
ChanCrd()
Pay rent
Buy brand
Roll double
Stop

B. Concepts

Concept of STL	Type	Code	Location(line)
Linked list	Link *	struct Link { int data; Link *LinkPtr; };	Link.h
		Link *bdName; Link *bdFront;	Player.h 25-26
Stack	Linked list	void addBdName(int); void addFront(int);	Player.h 38-39
		void pop(int);	Player.h 40
Find end	Linked list	Link *endLst(int);	Player.h 30
Search	Linked list	bool search(int, int);	Player.h 31
Print list	Linked list	void prntLst(int);	Player.h 44
Map	<int, int>	map<int, int>sell; map<int, int>coll;	Game.h 32-33
	<string, int>	map<string, int>value;	Game.h 35
	Iterator	map<string, int>::iterator iter;	Game.h 36

Vector	<int>	vector<int>score;	Game.h 37

Hash	string—>int	RSHash();
		BPHash();
		ELFHash();
			main.cpp 77-79

AVL Tree	Linked list	Self Balance Pre-order In-order Post-order	main.cpp 512-528

Quick Sort	int array[]	void valueSort();
		void quickSort(int [], int, int);
		int partition(int [], int, int);
		void qSwap(int &, int &);
			Game.cpp 99-150

C. Sample Input/Output

1. First output title

```
                                            [][][][][]
                                        [][][]      [][][]
[][][][][][][][][][][][][][][][][][][][][]              [][][][][][][][][][][][][][][][][][][][][]
[]                                                                                              []
[]  []          []    [][][]    []      []    [][][]    [][][]      [][][]    []    []      []  []
[]  [][]      [][]  []      []  [][]    []  []      []  []    []  []      []  []    []      []  []
[]  []  []  []  []  []      []  []  []  []  []      []  []    []  []      []  []      []  []    []
[]  []    []    []  []      []  []    [][]  []      []  [][][]    []      []  []        []      []
[]  []          []    [][][]    []      []    [][][]    []          [][][]    [][][]  []        []
[]                                                                                              []
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

Press enter to continue.
```

2. Register for user name and password

```
 Welcome to Monopoly: Empire!
 Please register your user name first (It's free!).

 User name: Sili
 Password: gsl
 Repeat password: gsl
```

3. Output main menu

```
 Monopoly: Empire

 Start Menu
 1. Start New Game
 2. Read Rules
 0. Quit
(For a better playing experience, strongly suggest read rules before you start the game!!!)

Please enter the number of your choice (0~2):
```

4. Recommend reading rules first

```
****************** Monopoly: Empire Rules ********************************
I. Basic rules
   A. Each player has 1000K at beginning and start at position #0.
   B. In each turn, player roll 2 dices, and move corresponding steps.
   C. When stop at a position, follow the instruction of that position.
   D. [1] represent player, [2] represent computer.

II. Game Board
   A. Total 24 positions (0~24) of 7 types.
   B. Start
       1. Every player start from here.
       2. When pass start point, collect as much money as your total collecting value.
   C. Billboard
       1. Containing name, price, collecting value, and condition(0,1 or 2).
       2. The first two lines are the name of the brand.
       3. The third line is price (collecting value), eg.100K (50).
```

```
            a. Price ≠ collecting value.
            b. Price is how much cost you to buy the brand.
            c. Collecting value is used for win the game.
        4. The fourth line is (condition), eg. (1).
             (0 = no owner; 1 = player own; 2 = computer own)
        5. The fifth line shows player's position, eg.[1] [2].
        6. The bottom line has the position # of that box.
        7. Exception: Electric Company Billboard. (See IV. Billboard-E)
    D. Chance Card
        1. Randomly pick a chance card when stop at here.
        2. Total 8 kinds of chance cards.
        3. Chance card may not be good for player who pick it.
    E. Jail
        1. To get out of jail, there are two choice.
            a. During next turn, pay 100K and get out at once.
            b. Roll a double up to 3 turns, if not, pay 50K and get out at 3rd turn.
        2. Position #19 is go to jail, player will be move to jail on Position #7.
        3. Simply stop at position #7 is pass by, is not in jail.
    F. Take a trip
        1. You can spend 100K to move to anywhere on board, or simply do nothing.
        2. If you move to a new position, you have to follow the instruction on that position.
    E. Tower Tax
        1. Tower Tax: return your topmost billboard to board.
        2. Rival Tower Tax: return your opponent's topmost billboard to board.

III. Dices
    A. One normal dice, one special dice with a 'swap' face.
    B. When 'swap' face is up, you have a chance of sneaky swap.
        (sneaky swap: swap your topmost billboard with your opponent's)
    C. When you roll a double, you can have another turn until you don't have a double.
        (double: two dices have same number)

IV. Billboard
    A. 14 billboards total; different price with different collecting value; choose smartly!
    B. You can buy billboards when you stop at the corresponding position if the condition is no owner.
    C. You have to pay rent fee if you stop at a position whose billboard is owner by your opponents.
    D. Nothing will happen if you stop at the position that is owned by yourself.
    E. One exception is Electric Company Billboard at position #9.
        1. Has 4 in total
        2. Doesn't own by anyone.

V. Chance Card
    A. Suggestion: Better read card function before playing game.
    B. Explanation of each card:
```

```
       1.Speed ahead!
Move forward to 5 space.
       2.Casino night!
Both you and your opponent roll.
Highest-roller collects 200K from the bank.
       3.Profits soar!
Advance to GO to collect your tower value.
       4.Launch your website!
Sales skyrocket!
Collect 300K from the bank.
       5.Solar power bonus!
Take a free Electric Company billboard.
Add it to your tower.
If none are available, do nothing.
       6.Insider trading fine!
Pay the Bank 200K.
       7.Tallest tower bonus!
Check what the highest tower is currently worth.
Collect that amount from the bank.
       8.Go To Jail!
Do not collect cash for passing GO.
```

```
VI. Win
   A. Whoever first get 800 collecting value wins.
   B. Whoever first run out of money loses.

VII. Bonus
   The bonus area will be open for player each time when player win a game.
VIII. Tips For Brand Buying
   A. The unit value for different brand is different. Be wise to choose which brand to buy!
   B. The brand unit value sorting (large -> small):

   TRANSFORMAERS >> PUMA >> GUITARHERO LIVE >> UNIVERSAL >> XBOX >> YAHOO!
    >> Candy Crush >> Ford >> LEVIS >> POLAROID >> RAZOR >> ebay >> skype >> ELECTRIC COMPANY

 Press enter to go back to Game Menu.
```

5. Start playing game, first enter password

```
Welcome, Sili!

To start game, please enter your password correctly:
gsl
```

6. Output game board

```
Monopoly: Empire Game Board:
 _____________________________________________________________________________________________________
|    FREE    | GUITARHERO |   CHANCE   |   YAHOO!   |    Ford    |   CHANCE   |    ebay    |   GO TO    |
|   PARKING  |    LIVE    |    CARD    |            |            |    CARD    |            |   JAIL!    |
|            | 250K (150) |            | 250K (150) | 300K (150) |            | 300K (150) |            |
|            |    (0)     |            |    (0)     |    (0)     |            |    (0)     |            |
|            |            |            |            |            |            |            |            |
|_____12_____|_____13_____|_____14_____|_____15_____|_____16_____|_____17_____|_____18_____|_____19_____|
|   Candy    |                                                                             | UNIVERSAL  |
|   Crush    |                                                                             |            |
| 200K (100) |                                                                             | 350K (200) |
|    (0)     |                                                                             |    (0)     |
|            |                                                                             |            |
|_____11_____|                                                                             |_____20_____|
|    LEVIS   |                                                                             |    XBOX    |
|            |                                                                             |            |
| 200K (100) |                                                                             | 350K (200) |
|    (0)     |                                                                             |    (0)     |
|            |                                                                             |            |
|_____10_____|                                                                             |_____21_____|
|  ELECTRIC  |                                                                             |   TOWER    |
|  COMPANY   |                                                                             |    TAX     |
| 150K(50,4) |                                                                             |            |
|            |                                                                             |            |
|            |                                                                             |            |
|_____09_____|                                                                             |_____22_____|
|    PUMA    |                                                                             |   skype    |
|            |                                                                             |            |
| 150K (100) |                                                                             | 400K (200) |
|    (0)     |                                                                             |    (0)     |
|            |                                                                             |            |
|_____08_____|_____________________________________________________________________________|_____23_____|
|  IN JAIL!  |  POLAROID  |   CHANCE   |    RAZOR   |   CHANCE   |TRANSFORMERS|   RIVAL    |    START   |
|__        __|            |    CARD    |            |    CARD    |            |   TOWER    |            |
|  PASS BY   | 100K (50)  |            | 100K (50)  |            |  50K (50)  |    TAX     |    GO!     |
|            |    (0)     |            |    (0)     |            |    (0)     |            |            |
|            |            |            |            |            |            |            |  [1]  [2]  |
|_____07_____|_____06_____|_____05_____|______04____|______03____|_____02_____|_____01_____|_____00_____|
```

7. Randomly choose who starts first

```
Randomly choosing who starts first...
You are first to start!
```

8. Rolling dices, and move corresponding steps

```
Rolling dice...
Your result is: 6 3
```

9. Asking if player would buy the brand

```
 Do you want to spend 150K to buy a Electric Company Billboard?

Please enter yes or no (y or n): y

You bought one Electric Company Billboard with 150K.
```

10. When stoping at 19, send to jail

```
You are sending to jail!

You are currently in jail, and you have two choices:
 1. Pay 100K to get out.
 2. Roll a double up to 3 turns; if not, pay 50K.

Please enter your choice (1 or 2): 
```

11.When stop at 3 or 17, pick a chance card

```
Computer randomly picks a chance card!
You pick Chance Card #4

-------------------------------------------
Solar power bonus!
Take a free Electric Company billboard.
Add it to your tower.
If none are available, do nothing.
-------------------------------------------

Computer collects a free Electric Company billboard.
```

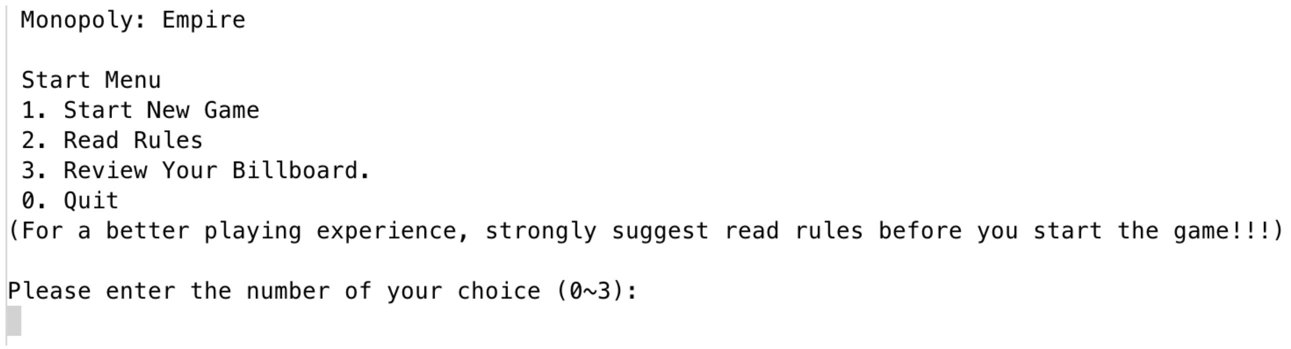
12. There will be player's info on the right side

```
Player's Info:
------------------------

Money: 150K
Brand Value:
50 200 50
------------------------

Money: 900K
Brand Value:
50 150 50
```

13. Player lose when money goes to negative

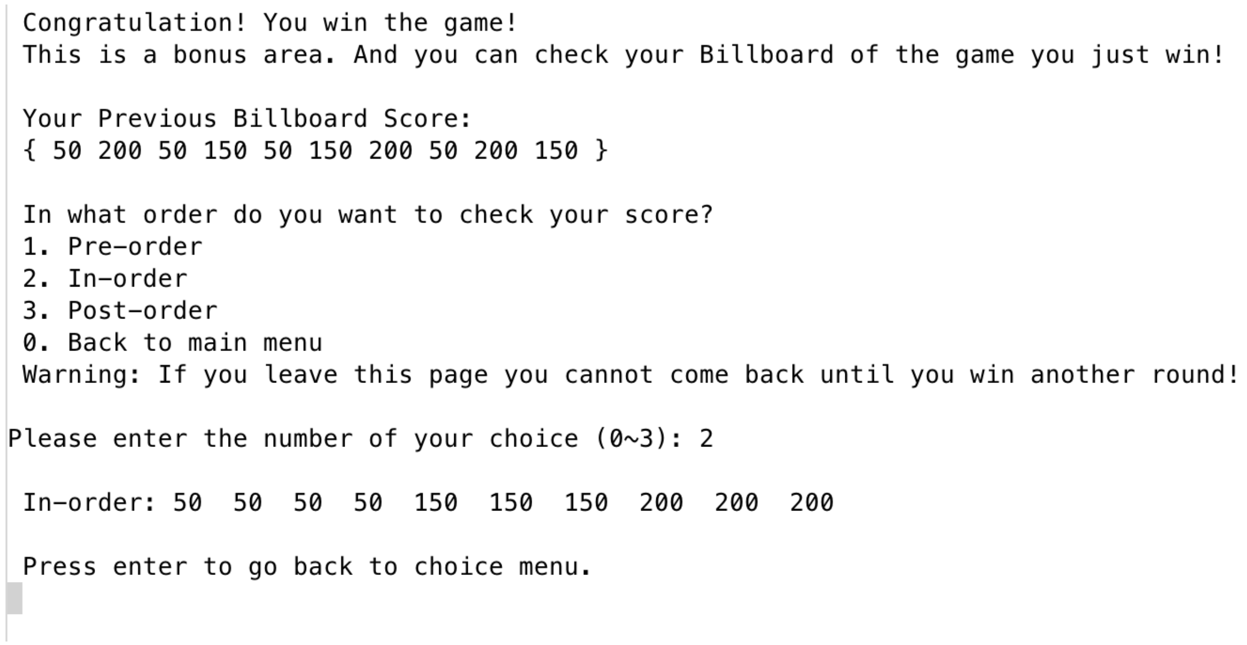
```
Money: -150K
```

14. Player win when billboard reach 800, then bonus area will appear



```
 Monopoly: Empire

 Start Menu
 1. Start New Game
 2. Read Rules
 3. Review Your Billboard.
 0. Quit
(For a better playing experience, strongly suggest read rules before you start the game!!!)

Please enter the number of your choice (0~3):
```

15. Bonus are



```
 Congratulation! You win the game!
 This is a bonus area. And you can check your Billboard of the game you just win!

 Your Previous Billboard Score:
 { 50 200 50 150 50 150 200 50 200 150 }

 In what order do you want to check your score?
 1. Pre-order
 2. In-order
 3. Post-order
 0. Back to main menu
 Warning: If you leave this page you cannot come back until you win another round!

Please enter the number of your choice (0~3): 2

 In-order: 50  50  50  50  150  150  150  200  200  200

 Press enter to go back to choice menu.
```

VI. Code

main.cpp

```
/*
 * File:   main.cpp
 * Author: Sili Guo
 * Created on April 14, 2017
 * Purpose: Game---Monopoly: Empire
 */


//System Libraries
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cctype>
using namespace std;


//User Libraries
#include "Game.h"
#include "GeneralHashFunctions.h"
#include "AVLTree.h"
//Global Constants
```

```
//Function Prototypes
void enterRcd(char);


//Execution Begins here!


int main(int argc, char** argv) {
    //Random number seed
    srand(static_cast<unsigned int> (time(0)));
    //Declare Variables
    Game game; //New game
    char enter; //Record enter
    char cho; //Record player's choice
    int choice; //Transfer cho to int for calculation
    string pName; //Player's name
    string pass; //Player's password
    string repeat; //Repeat password
    string inputPass; //User input password
    string msg; //Message for hash
    string hash1; //Hash of message
    string hash2; //Hash of password
    string test2; //Hash of test password
```

```
int Hash; //Hash of hash1 and hash2

int Test; //Hash result of test

char sneaky; //Answer for sneaky swap

int pos; //Hold the position info

char buy; //Hold buying info

char jum; //Hold answer for jump

int jail; //choice in jail

int start; //start the game first

bool doub; //Judge for rolling double

//Title

game.title();

enterRcd(enter);

//Register for user name and password

do {

   //Next page

   system("clear");

   cout << "\n Welcome to Monopoly: Empire!" << endl;

   cout << " Please register your user name first (It's free!). " << endl;

   cout << "\n User name: ";

   cin >> pName;

   game.setName(pName, "Computer");

   cout << " Password: ";
```

```
    cin >> pass;
    cout << " Repeat password: ";
    cin >> repeat;
    //If two input didn't match, ask for reenter
    if (repeat != pass) {
        cout << "\n Error! Your repeat password is different with your password!" << endl;
        cout << " Press enter to refill your information." << endl;
        cin.ignore();
        enterRcd(enter);
    }
} while (repeat != pass);
//Hash password
msg = "This program is awesome!";
hash1 = to_string(RSHash(msg));
hash2 = to_string(BPHash(pass));
Hash = ELFHash(hash1 + hash2);
//Game main menu
do {
    system("clear");
    cout << "\n Monopoly: Empire" << endl;
    cout << "\n Start Menu" << endl;
    cout << " 1. Start New Game" << endl;
    cout << " 2. Read Rules" << endl;
    //After player win
    if (game.getWin()) cout << " 3. Review Your Billboard." << endl;
    cout << " 0. Quit" << endl;
    cout << "(For a better playing experience, strongly suggest read rules before you start the
game!!!)" << endl;
    do {
      if (game.getWin()) cout << "\nPlease enter the number of your choice (0~3):" << endl;
      else cout << "\nPlease enter the number of your choice (0~2):" << endl;
      cin >> cho;
      if (!game.getWin() && cho == 51) cho = 52;
      if (cho != 48 && cho != 49 && cho != 50 && cho != 51) {
        if (game.getWin()) cout << "Invalid input! Please enter a number between 0 to 3." <<
endl;
        else cout << "Invalid input! Please enter a number between 0 to 2." << endl;
      }
    } while (cho != 48 && cho != 49 && cho != 50 && cho != 51);
    //Transfer cho to int
    choice = cho - 48;
    //Reset initial condition
    game.reset();
    //Quit the game
```

```
if (choice != 1 && choice != 2 && choice != 3) {
  system("clear");
  cout << "You quit the game." << endl;
  break;
}
switch (choice) {
  case 1:
    do {
      system("clear");
      //Promote for players' name
      cout << "\n Welcome, " << pName << "!" << endl;
      cout << "\n To start game, please enter your password correctly:" << endl;
      cout << " ";
      cin >> inputPass;
      //Hash new input password
      test2 = to_string(BPHash(inputPass));
      Test = ELFHash(hash1 + test2);
      //Check password
      if (Test != Hash) {
        cout << "\n Sorry, you enter the wrong password! Please try again!" << endl;
        cin.ignore();
        enterRcd(enter);
        }
    } while (Test != Hash);
    system("clear");
    //Output the game board
    cout << "Monopoly: Empire Game Board:" << endl;
    game.output();
    //Randomly choose who starts first
    cout << "\nRandomly choosing who starts first..." << endl;
    game.setStart();
    start = game.getStart();
    if (start == 0) cout << "You are first to start!" << endl;
    else cout << "Computer starts first." << endl;
    //Next page
    cout << "\nPress enter to continue." << endl;
    cin.ignore();
    enterRcd(enter);
    do {
        //Turn 1
        system("clear");
        cout << "\n Turn #" << game.getTurn() << endl;
        game.output();
        switch (start) {
```

```
case 0://Player starts first
    //If player is in jail
    do {
        doub = false;
        cout << "Your turn:" << endl;
        if (game.getJail(1) == true) {
            cout << "\nYou are currently in jail, and you have two choices: " << endl;
            cout << " 1. Pay 100K to get out." << endl;
            cout << " 2. Roll a double up to 3 turns; if not, pay 50K." << endl;
            do {
                cout << "\nPlease enter your choice (1 or 2): ";
                cin >> jail;
                cout << endl;
                if (jail < 1 || jail > 2)
                    cout << "Invalid input! Please enter 1 or 2." << endl;
                if (game.jailJudge(jail) == false)
                    cout << "You are unable to pay the fee." << endl;
            } while (jail < 1 || jail > 2 || game.jailJudge(jail) == false);
            game.outJail(1, jail);
            //Next page
            cout << "\nPress enter to continue." << endl;
            cin.ignore();
```

```
                        enterRcd(enter);
                        //Output
                        game.output();
                    }
                    if (game.getJail(1) == false) {
                        //First roll two dice
                        cout << "\nRolling dice..." << endl;
                        game.setDices();
                        cout << "Your result is: " << game.getDice1() << " " << game.getDice2()
<< endl << endl;
                        if (game.getDice1() == game.getDice2()) {
                            doub = true;
                            cout << "Congratulation! You rolled double and earned an extra turn."
<< endl;
                        }
                        //If get the swap face up
                        if (game.getDice1() == 's') {
                            cout << "Do you want to use the sneaky swap?" << endl;
                            cout << "(Switch your topmost billboard with your opponent's topmost
billboard," << endl;
                            cout << "then don't move this turn; or just use the second dice to
move.)" << endl;
```

```
    do {
        cout << "\nPlease enter yes or no (y or n): ";
        cin >> sneaky;
        if (tolower(sneaky) != 'y' && tolower(sneaky) != 'n')
            cout << "Invalid input! Please enter y or n." << endl;
    } while (tolower(sneaky) != 'y' && tolower(sneaky) != 'n');
    //Do sneaky swap
    if (tolower(sneaky) == 'y') {
        cout << "\nYou use sneaky swap." << endl;
        game.swap();
        //Or just use the second dice
    } else {
        cout << "\nYou did not use sneaky swap." << endl;
        game.move(1, game.getDice2() - 48);
    }
    cout << "\nPress enter to continue." << endl;
    cin.ignore();
} else {//Normally just move by using two dices
    game.move(1, (game.getDice1() - 48) + (game.getDice2() - 48));
    cout << "\nPress enter to continue." << endl;
}
enterRcd(enter);
```

```
//Output
game.output();
//Ask player to buy brand if land on it
pos = game.getPos(1);
if (pos == 12) {
    cout << "\nDo you want to take a trip (100K)?" << endl;
    cout << "(You can spend 100K to move to anywhere on board or do
nothing)" << endl;
    do {
        cout << "\nPlease enter yes or no (y or n): ";
        cin >> jum;
        cout << endl;
        if (tolower(jum) != 'y' && tolower(jum) != 'n')
            cout << "Invalid input! Please enter y or n." << endl;
    } while (tolower(jum) != 'y' && tolower(jum) != 'n');
    if (tolower(jum) == 'y') {
        cout << "Where do you want to go?" << endl;
        do {
            cout << "Please enter the position # (0~23): ";
            cin >> pos;
            cout << endl;
            if (pos < 0 || pos > 23 || pos == 12)
```

```
                    cout << "Invalid input! Please enter a number between 0~23."
<< endl;
                } while (pos < 0 || pos > 23 || pos == 12);
                game.jump(1, pos);
                cout << "\nYou jump to position #" << pos << endl;
            }
            //Next page
            cout << "\nPress enter to continue." << endl;
            cin.ignore();
            enterRcd(enter);
            //Output
            game.output();
        }
        if (pos == 2 || pos == 4 || pos == 6 || pos == 8 || pos == 10 || pos == 11 ||
pos == 13
                || pos == 15 || pos == 16 || pos == 18 || pos == 20 || pos == 21 || pos
== 23) {
            cout << game.sale(pos) << endl;
            if (game.sale(pos) == 0) {
                cout << "\nYou stopped at position #" << pos << endl;
                cout << "Do you want to spend " << game.getSell(pos) << "K to buy
this brand?" << endl;
```

```
                    do {
                        cout << "\nPlease enter yes or no (y or n): ";
                        cin >> buy;
                        cout << endl;
                        if (tolower(buy) != 'y' && tolower(buy) != 'n')
                            cout << "Invalid input! Please enter y or n." << endl;
                    } while (tolower(buy) != 'y' && tolower(buy) != 'n');
                    //Buy the brand
                    if (tolower(buy) == 'y') {
                        cout << "You bought the brand at position " << pos << " with " <<
game.getSell(pos) << "K." << endl;
                        game.buyBrnd(1, pos);
                    }
                    cin.ignore();
                } else if (game.sale(pos) == 2) {
                    if (game.getJail(2)) {
                        cout << "\nComputer is currently in jail." << endl;
                        cout << "You don't need to pay the fee." << endl;
                    } else {
                        cout << "\nThis brand belongs to " << game.getName(2) << "." <<
endl;
```

```
                    cout << "You have to pay " << game.getName(2) << " " <<
game.getSell(pos) << "K." << endl;
                    game.payRent(1, pos);
                  }
                } else cout << "\nYou stop at the brand belongs to you." << endl;
                //Next page
                cout << "\nPress enter to continue." << endl;
                enterRcd(enter);
                //Output
                game.output();
              } else if (pos == 19) {
                cout << "\nYou are sending to jail!" << endl;
                game.jail(1);
                //Next page
                cout << "\nPress enter to continue." << endl;
                enterRcd(enter);
                //Output
                game.output();
              } else if (pos == 1) {
                cout << "\nYou stopped at Rival Tower Tax." << endl;
                cout << "You return Computer's topmost billboard to the board." <<
endl << endl;
```

```
            game.remove(2);
            //Next page
            cout << "\nPress enter to continue." << endl;
            enterRcd(enter);
            //Output
            game.output();
        } else if (pos == 22) {
            cout << "\nYou stopped at Tower Tax." << endl;
            cout << "You return your topmost billboard to the board." << endl <<
endl;
            game.remove(1);
            //Next page
            cout << "\nPress enter to continue." << endl;
            enterRcd(enter);
            //Output
            game.output();
        } else if (pos == 3 || pos == 5 || pos == 14 || pos == 17) {
            cout << "\nYou randomly pick a chance card!" << endl;
            game.chanCrd(1);
            //Next page
            cout << "\nPress enter to continue." << endl;
            enterRcd(enter);
```

```
                    //Output
                    game.output();
                  } else if (pos == 9) {
                    if (game.getElc() > 0) {
                      cout << "\n Do you want to spend 150K to buy a Electric Company
Billboard?" << endl;
                      do {
                        cout << "\nPlease enter yes or no (y or n): ";
                        cin >> buy;
                        cout << endl;
                        if (tolower(buy) != 'y' && tolower(buy) != 'n')
                          cout << "Invalid input! Please enter y or n." << endl;
                      } while (tolower(buy) != 'y' && tolower(buy) != 'n');
                      if (tolower(buy) == 'y') {
                        cout << "You bought one Electric Company Billboard with " <<
game.getSell(pos) << "K." << endl;
                        game.buyElec(1);
                      }
                      cin.ignore();
                    } else cout << "\nNo Electric Company Billboard available!" << endl;
                    //Next page
                    cout << "\nPress enter to continue." << endl;
```

```
                enterRcd(enter);
                //Output
                game.output();
            }
        }
    } while (doub);
    game.setOver();
    if (game.getOver()) break;
case 1:
    do {
        doub = false;
        cout << "Computer's turn: " << endl;
        //If player is in jail
        if (game.getJail(2) == true) {
            cout << "\nComputer is currently in jail." << endl << endl;
            game.outJail(2, 0);
            //Next page
            cout << "\nPress enter to continue." << endl;
            enterRcd(enter);
            //Output
            game.output();
        }
```

```
                if (game.getJail(2) == false) {
                    //First roll two dices
                    cout << "\nRolling dice..." << endl;
                    game.setDices();
                    cout << "Computer's result is: " << game.getDice1() << " " << game.get-
Dice2() << endl << endl;
                    if (game.getDice1() == game.getDice2()) {
                        doub = true;
                        cout << "Computer rolled double and earned an extra turn.";
                    }
                    //If get the swap face up
                    if (game.getDice1() == 's') {
                        if (game.compare()) {
                            cout << "Computer used sneaky swap!" << endl;
                            game.swap();
                        } else {
                            cout << "Computer didn't use sneaky swap!" << endl;
                            game.move(2, game.getDice2() - 48);
                        }
                    } else {
                        game.move(2, (game.getDice1() - 48) + (game.getDice2() - 48));
                    }
```

```
//Next page
cout << "\nPress enter to continue." << endl;
enterRcd(enter);
//Output
game.output();
//Ask player to buy brand if land on it
pos = game.getPos(2);
if (pos == 12) {
    game.tripAI();
    //Next page
    cout << "\nPress enter to continue." << endl;
    enterRcd(enter);
    //Output
    game.output();
}
if (pos == 2 || pos == 4 || pos == 6 || pos == 8 || pos == 10 || pos == 11 ||
pos == 13
        || pos == 15 || pos == 16 || pos == 18 || pos == 20 || pos == 21 || pos
== 23) {
    if (game.sale(pos) == 0) {
        if (game.getMoney(2) > 500) {
```

```
                    cout << "\nComputer bought the brand at position " << pos << "
with " << game.getSell(pos) << "K." << endl;
                    game.buyBrnd(2, pos);
                  } else if (game.getMoney(2) > 2 * game.getSell(pos)) {
                    cout << "\nComputer bought the brand at position " << pos << "
with " << game.getSell(pos) << "K." << endl;
                    game.buyBrnd(2, pos);
                  }
                } else if (game.sale(pos) == 1) {
                  if (game.getJail(1)) {
                    cout << "\nYou are currently in jail." << endl;
                    cout << "Computer did not pay you." << endl;
                  } else {
                    cout << "\nComputer stop at the brand belongs to you." << endl;
                    cout << "Computer paid you " << game.getSell(pos) << "K." <<
endl;
                    game.payRent(2, pos);
                  }
                }
                //Next page
                cout << "\nPress enter to continue." << endl;
                enterRcd(enter);
```

```
          //Output
          game.output();
        } else if (pos == 19) {
          cout << "\nComputer is sending to jail!" << endl;
          game.jail(2);
          //Next page
          cout << "\nPress enter to continue." << endl;
          enterRcd(enter);
          //Output
          game.output();
        } else if (pos == 1) {
          cout << "\nComputer stopped at Rival Tower Tax." << endl;
          cout << "Computer return your topmost billboard to the board." << endl
<< endl;
          game.remove(1);
          //Next page
          cout << "\nPress enter to continue." << endl;
          enterRcd(enter);
          //Output
          game.output();
        } else if (pos == 22) {
          cout << "\nComputer stopped at Tower Tax." << endl;
```

```
                    cout << "Computer return its topmost billboard to the board." << endl
<< endl;
                    game.remove(2);
                    //Next page
                    cout << "\nPress enter to continue." << endl;
                    enterRcd(enter);
                    //Output
                    game.output();
                } else if (pos == 3 || pos == 5 || pos == 14 || pos == 17) {
                    cout << "\nComputer randomly picks a chance card!" << endl;
                    game.chanCrd(2);
                    //Next page
                    cout << "\nPress enter to continue." << endl;
                    enterRcd(enter);
                    //Output
                    game.output();
                } else if (pos == 9) {
                    if (game.getElc() > 0) {
                        if (game.getMoney(2) > 500) {
                            cout << "\nComputer bought one Electric Company Billboard with
" << game.getSell(pos) << "K." << endl;
                            game.buyElec(2);
                        } else if (game.getMoney(2) > 2 * game.getSell(pos)) {
                            cout << "\nComputer bought one Electric Company Billboard with
" << game.getSell(pos) << "K." << endl;
                            game.buyElec(2);
                        }
                    }
                    //Next page
                    cout << "\nPress enter to continue." << endl;
                    enterRcd(enter);
                    //Output
                    game.output();
                }
            }
        } while (doub);
        game.setOver();
        if (game.getOver()) break;
    }//End of switch (start first)
    start = 0;
} while (game.getOver() == false);
//Record Game
game.record();
//Next page
```

```
        cout << "\nPress enter to continue." << endl;

        enterRcd(enter);

        //Output

        game.output();

        if (game.getWin() == true) {

            cout << "\n Congratulation! You win the game." << endl;

        } else {

            cout << "\n Sorry you lose the game." << endl;

        }

        break;

    case 2:

        system("clear");

        game.rules();

        break;

    case 3:

        do {

            do {

                system("clear");

                cout << "\n Congratulation! You win the game!" << endl;

                cout << " This is a bonus area. And you can check your Billboard of the game you
just win!" << endl;

                cout << "\n Your Previous Billboard Score: " << endl;
            game.pntScore();
            cout << "\n In what order do you want to check your score? " << endl;
            cout << " 1. Pre-order" << endl;
            cout << " 2. In-order" << endl;
            cout << " 3. Post-order" << endl;
            cout << " 0. Back to main menu" << endl;
            cout << " Warning: If you leave this page you cannot come back until you win
another round!" << endl;
            cout << "\nPlease enter the number of your choice (0~3): ";
            cin >> choice;
            //check validation
            if (choice != 0 && choice != 1 && choice != 2 && choice != 3) {
                cout << "\nInvalid Input! Please enter a number between 0 to 3." << endl;
                cin.ignore();
                enterRcd(enter);
            }
        } while (choice != 0 && choice != 1 && choice != 2 && choice != 3);
        if (choice != 1 && choice != 2 && choice != 3) break;
        AVLTree tree;
        for (int i = 0; i < game.getScoreSize(); i++) {
            tree.root = tree.insert(tree.root, game.getScore(i));
        }
```

```
            switch (choice) {
                case 1:
                    cout << "\n Pre-order: ";
                    tree.preorder(tree.root);
                    break;
                case 2:
                    cout << "\n In-order: ";
                    tree.inorder(tree.root);
                    break;
                case 3:
                    cout << "\n Post-order: ";
                    tree.postorder(tree.root);
            }
            cout << "\n\n Press enter to go back to choice menu." << endl;
            cin.ignore();
            do {
                enter = getchar();
            } while (enter != '\n');
        } while (enter == '\n');
        break;
}//End of switch (game menu)
cout << "\n Press enter to go back to Game Menu." << endl;
```

```
        if (choice == 2 || choice == 3) cin.ignore();
        do {
            enter = getchar();
        } while (enter != '\n');
    } while (enter == '\n');
    //Exit Stage right!
    return 0;
}

void enterRcd(char en) {
    do {
        en = getchar();
    } while (en != '\n');
    system("clear");
}
```

Link.h

```
/*
 * File:   Link.h
 * Author: Sili Guo
 * Created on April 14, 2017
 * Purpose: Game---Monopoly: Empire; structure for linked list
 */

#ifndef LINK_H
#define LINK_H

struct Link {
    int data;
    Link *LinkPtr;
};
#endif /* LINK_H */
```

Player.h

```
/*
 * File:   Player.h
 * Author: Sili Guo
 * Created on April 14, 2017
 * Purpose: Game---Monopoly: Empire
 */

#ifndef PLAYER_H
#define PLAYER_H
```

```
#include <iostream>

using namespace std;


#include "Link.h"


class Player {

private:

    string name; //Player's name

    string password;//Player's password

    int money; //Money in player's hand

    int position; //Current position on board

    int bonus; //Bonus player totally collected

    bool jail; //Condition in jail

    int jTurn; //Turns in jail;

    Link *bdName; //The front of Billboard Titles names

    Link *bdFront; //The front of Billboard Titles linked list

    int sum; //Billboard collecting value sum

public:

    Player(); //Constructor

    virtual ~Player(); //Destructor

    void reset();

    Link *endLst(int);
```

```
bool search(int, int);
void setName(string);
void setPosition(int);
void setJump(int);
void setSum();
void addBdName(int);
void addBdFront(int);
void sptMoney(int);
void takMoney(int);
void pop(int);
void inJail();
void oJail();
void jailTurn();
void prntLst(int);


string getName() {
   return name;
}


int getMoney() {
   return money;
}
```

```
int getPostion() {
    return position;
}

bool getJail() {
    return jail;
}

int getJTurn() {
    return jTurn;
}

Link *getBdName() {
    return bdName;
}

Link *getBdFront() {
    return bdFront;
}

int getSum() {
        return sum;
    }
};

#endif /* PLAYER_H */
```

Player.cpp

```
/*
 * File:   Player.cpp
 * Author: Sili Guo
 * Created on April 14, 2017
 * Purpose: Game---Monopoly: Empire
 */

#include "Player.h"

Player::Player() {
    name = " "; //Initialize name to empty
    password = " "; //Initialize password to empty
    money = 1000; //Initialize money to 800K
    position = 0; //Initialize position at starting point
    bonus = 0; //Initialize total bonus to 0
    jail = false; //Not in jail

    jTurn = 0; //Initialize turns in jail to 0

    bdName = NULL; //Have no Billboard Titles

    bdFront = NULL; //Have no Billboard Titles

    sum = 0;

}


Player::~Player() {

    if (bdName == NULL && bdFront == NULL) return;

    if (bdName != NULL) {

        do {

            Link *temp = bdName->LinkPtr;

            delete bdName;

            bdName = temp;

        } while (bdName != NULL);

    }

    if (bdFront != NULL) {

        do {

            Link *temp = bdFront->LinkPtr;

            delete bdFront;

            bdFront = temp;

        } while (bdFront != NULL);
    }
}

void Player::reset() {
    name = " "; //Initialize name to empty
    password = " "; //Initialize password to empty
    money = 1000; //Initialize money to 800K
    position = 0; //Initialize position at starting point
    bonus = 0; //Initialize total bonus to 0
    jail = false; //Not in jail
    jTurn = 0; //Initialize turns in jail to 0
    bdName = NULL; //Have no Billboard Titles
    bdFront = NULL; //Have no Billboard Titles
    sum = 0;
}

Link *Player::endLst(int n) {
    Link *lnk, *end;
    if (n == 1) {
        if (bdName == NULL) end = NULL;
        else {
            lnk = bdName;
            do {
                end = lnk; //Are we at the end yet?
                lnk = lnk->LinkPtr; //Traverse to the next link
            } while (lnk != NULL); //Finally found the end when NULL
        }
    } else if (n == 2) {
        if (bdFront == NULL) end = NULL;
        else {
            lnk = bdFront;
            do {
                end = lnk; //Are we at the end yet?
                lnk = lnk->LinkPtr; //Traverse to the next link
            } while (lnk != NULL); //Finally found the end when NULL
        }
    }
    return end;
}

bool Player::search(int s, int n) {
    if (s == 1) {
        if (bdName == NULL) return false;
        Link *lnk = bdName;
        while (lnk->data != n) {
            lnk = lnk->LinkPtr;
            if (lnk == NULL) return false;
        }
    } else if (s == 2) {
        if (bdFront == NULL) return false;
        Link *lnk = bdFront;
        while (lnk->data != n) {
            lnk = lnk->LinkPtr;
            if (lnk == NULL) return false;
        }
    }
    return true;
}

void Player::setName(string n) {
    name = n;
}

void Player::setPosition(int n) {
    position += n;
    if (position > 23) {
        position -= 24;

        setSum();

        money += sum;

        cout << name << " passed start point and collect " << sum << "K from bank." << endl;

    }

}


void Player::setJump(int n) {

    position = n;

}


void Player::setSum() {

    sum = 0;

    Link *lnk = bdFront;

    while (lnk != NULL) {

        sum += lnk->data;

        lnk = lnk->LinkPtr;

    }

}


void Player::inJail() {

    //    if (jail == true) jail = false;
    //    if (jail == false) jail = true;
    jail = true;
}

void Player::oJail() {
    jail = false;
    position = 7;
}

void Player::jailTurn() {
    if (jTurn >= 0 && jTurn <= 3) jTurn++;
    else return;
}

void Player::addBdName(int n) {
    Link *lnk = new Link;
    lnk->data = n;
    lnk->LinkPtr = NULL;
    if (bdName == NULL) {
        bdName = lnk;
    } else {
        Link *end = endLst(1);
        if (end != NULL) end->LinkPtr = lnk;
    }
}

void Player::addBdFront(int n) {
    Link *lnk = new Link;
    lnk->data = n;
    lnk->LinkPtr = NULL;
    if (bdFront == NULL) {
        bdFront = lnk;
    } else {
        Link *end = endLst(2);
        if (end != NULL) end->LinkPtr = lnk;
    }
}

void Player::pop(int n) {
    if (n == 1) {
        if (bdName == NULL) return;
        else if (bdName->LinkPtr == NULL)
            bdName = NULL;
        else {
            Link *lnk = bdName;
            Link *end = endLst(1);
            while (lnk->LinkPtr != end)
                lnk = lnk->LinkPtr;
            lnk->LinkPtr = NULL;
        }//End of else
    } else if (n == 2) {
        if (bdFront == NULL) return;
        else if (bdFront->LinkPtr == NULL)
            bdFront = NULL;
        else {
            Link *lnk = bdFront;
            Link *end = endLst(2);
            while (lnk->LinkPtr != end)
                lnk = lnk->LinkPtr;
            lnk->LinkPtr = NULL;
        }//End of else
    }
}

void Player::sptMoney(int m) {
    money -= m;
```

```
}

void Player::takMoney(int m) {
   money += m;
}

void Player::prntLst(int n) {
   if (n == 1) {
      if (bdName == NULL) {
         cout << " ";
         return;
      }
      Link *next = bdName;
      do {
         cout << next->data << " ";
         next = next->LinkPtr;
      } while (next != NULL);
   } else if (n == 2) {
      if (bdFront == NULL) {
         cout << " " << endl;
         return;
      }
```

```
        Link *next = bdFront;

        do {

            cout << next->data << " ";

            next = next->LinkPtr;

        } while (next != NULL);

    }

    cout << endl;

}
```

Game.h

```
/*

 * File:   Player.h

 * Author: Sili Guo

 * Created on April 14, 2017

 * Purpose: Game---Monopoly: Empire

 */


#ifndef GAME_H

#define GAME_H


#include <iostream>

#include <cstdlib>
```

```
#include <iomanip>
#include <map>
#include <vector>
using namespace std;

#include "Player.h"

class Game {
private:
    int turn; //Game turns
    int cond; //Billboard condition
    int start; //Who starts first
    int elcNum; //Number of electric company
    bool over; //Judge for game over
    bool win; //If player wins
    int *chance; //8 Chance Card
    Player p1; //First player
    Player p2; //Second player
    char dice1; //First dice: the special one
    char dice2; //Second dice: the normal one
    map<int, int>sell; //Brand selling price
    map<int, int>coll; //Brand collecting value
    map<string, int>value; //Unit value for each brand
    map<string, int>::iterator iter; //Iterator for unit value
    vector<int>score; //Record player's Billboard
    bool winLose; //Record if player win
    void initial();
public:
    Game();
    virtual ~Game();
    void reset();
    void record();
    void valueSort();
    void quickSort(int [], int, int);
    int partition(int [], int, int);
    void qSwap(int &, int &);
    int sale(int);
    void shuffle(int *);
    string stopAt(int);
    string defineC(int);
    void chanCrd(int);
    void swap();
    void move(int, int);
    void jump(int, int);
```

```
bool compare();

void tripAI();

void jail(int);

bool jailJudge(int);

void outJail(int, int);

void buyBrnd(int, int);

void buyElec(int);

void payRent(int, int);

void remove(int);

void title();

void output();

void pntScore();

void setOver();

void setName(string, string);

void setStart();

void setDices();

int getPos(int);

int getMoney(int);

string getName(int);

void rules();


char getDice1() {
    return dice1;
}

char getDice2() {
    return dice2;
}

int getStart() {
    return start;
}

int getTurn() {
    return ++turn;
}

int *getChance() {
    return chance;
}

int getSell(int p) {
    return sell[p];
}
```

```
int getColl(int p) {
    return coll[p];
}
int getJail(int);


bool getOver() {
    return over;
}


bool getWin() {
    return win;
}


int getElc() {
    return elcNum;
}


int getScore(int n) {
    return score[n];
}
```

```
    int getScoreSize() {
        return score.size();
    }

    int getWinLose() {
        return winLose;
    }
};

#endif /* GAME_H */
```

Game.cpp

```
/*
 * File:   Player.h
 * Author: Sili Guo
 * Created on April 14, 2017
 * Purpose: Game---Monopoly: Empire
 */

#include "Game.h"

Game::Game() {
    turn = 0; //Initialize game turns to 1
    cond = 0; //Initialize all Billboard Titles are not sold
    start = 0; //Initialize start to 0;
    elcNum = 4; //Four electric company in total
    over = false; //Game not over
    win = false; //Player not win
    chance = new int[8];
    for (int i = 0; i < 8; i++)
        chance[i] = i;
    initial();
}

void Game::initial() {
    //Selling price
    sell[2] = 50;
    sell[4] = 100;
    sell[6] = 100;
    sell[8] = 150;
    sell[9] = 150;
    sell[10] = 200;
    sell[11] = 200;
    sell[13] = 250;
```

```
sell[15] = 250;

sell[16] = 300;

sell[18] = 300;

sell[20] = 350;

sell[21] = 350;

sell[23] = 400;

//Collecting value

coll[2] = 50;

coll[4] = 50;

coll[6] = 50;

coll[8] = 100;

coll[9] = 50;

coll[10] = 100;

coll[11] = 100;

coll[13] = 150;

coll[15] = 150;

coll[16] = 150;

coll[18] = 150;

coll[20] = 200;

coll[21] = 200;

coll[23] = 200;

//Unit value
```

```
    value["TRANSFORMAERS"] = 6 * sell[2] / coll[2];

    value["RAZOR"] = 6 * sell[4] / coll[4];

    value["POLAROID"] = 6 * sell[6] / coll[6];

    value["PUMA"] = 6 * sell[8] / coll[8];

    value["ELECTRIC COMPANY"] = 6 * sell[9] / coll[9];

    value["LEVIS"] = 6 * sell[10] / coll[10];

    value["Candy Crush"] = 6 * sell[11] / coll[11];

    value["GUITARHERO LIVE"] = 6 * sell[13] / coll[13];

    value["YAHOO!"] = 6 * sell[15] / coll[15];

    value["Ford"] = 6 * sell[16] / coll[16];

    value["ebay"] = 6 * sell[18] / coll[18];

    value["UNIVERSAL"] = 6 * sell[20] / coll[20];

    value["XBOX"] = 6 * sell[21] / coll[21];

    value["skype"] = 6 * sell[23] / coll[23];

}


void Game::reset() {

    turn = 0; //Initialize game turns to 1

    cond = 0; //Initialize all Billboard Titles are not sold

    start = 0; //Initialize start to 0;

    elcNum = 4; //Four electric company in total

    over = false; //Game not over
    win = false; //Player not win
    p1.reset();
    p2.reset();
}

void Game::record() {
    winLose = getWin();
    Link *temp = p1.getBdFront();
    while (temp != NULL) {
        score.push_back(temp->data);
        temp = temp->LinkPtr;
    }
}

void Game::pntScore() {
    cout << " { ";
    for (int i = 0; i < score.size(); i++) {
        cout << score[i] << " ";
    }
    cout << "}" << endl;
}
```

```
void Game::valueSort() {
    int array[14];
    iter = value.begin();
    for (int i = 0; i < 14; i++) {
        array[i] = iter->second;
        iter++;
    }
    quickSort(array, 0, 13);
    for (iter = value.begin(); iter != value.end(); iter++) {
        if (array[0] == iter->second) {
            cout << "    " << iter->first;
        }
    }
    for (int i = 1; i < 14; i++) {
        if (array[i] != array[i - 1]) {
            for (iter = value.begin(); iter != value.end(); iter++) {
                if (array[i] == iter->second)
                    cout << " >> " << iter->first;
            }
        }
        if (i == 5) cout << endl << "    ";
    }
```

```
}

void Game::quickSort(int a[], int p, int r) {
    int q;
    if (p < r) {
        q = partition(a, p, r);
        quickSort(a, p, q - 1);
        quickSort(a, q + 1, r);
    }
}

int Game::partition(int a[], int p, int r) {
    int x = a[r];
    int i = p - 1;
    for (int j = p; j < r; j++) {
        if (a[j] <= x) {
            i++;
            qSwap(a[i], a[j]);
        }
    }
    qSwap(a[i + 1], a[r]);
    return i + 1;
}

void Game::qSwap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

int Game::sale(int n) {
    if (p1.search(1, n) == true) cond = 1;
    else if (p2.search(1, n) == true) cond = 2;
    else cond = 0;
    return cond;
}

Game::~Game() {
    delete [] chance;
}

void Game::setName(string n1, string n2) {
    p1.setName(n1);
    p2.setName(n2);
}

void Game::setStart() {
    start = rand() % 2;
    //    start = 0;
}

void Game::setDices() {
    dice1 = rand() % 6 + 49;
    if (dice1 == 49)
        dice1 = 's';
    dice2 = rand() % 6 + 49;
}

void Game::setOver() {
    if (p1.getMoney() <= 0) {
        cout << "You run out of money!" << endl;
        over = true;
        win = false;
    }
    if (p2.getMoney() <= 0) {
        cout << "Computer runs out of money!" << endl;
        over = true;
        win = true;
    }
    p1.setSum();
    if (p1.getSum() >= 800) {
        cout << "Your billboard reach the top!" << endl;
        over = true;
        win = true;
    }
    p2.setSum();
    if (p2.getSum() >= 800) {
        cout << "Computer's billboard reach the top!" << endl;
        over = true;
        win = false;
    }
}

string Game::stopAt(int n) {
    string pos;
    if (p1.getPostion() == n)
        pos = "[1] ";
    else pos = "    ";

    if (p2.getPostion() == n)

        pos += " [2]";

    else pos += "    ";

    return pos;

}


void Game::shuffle(int *card) {

    int temp;

    int r; //Hold a random number

    for (int i = 0; i < 8; i++) {

        r = rand() % 8;

        temp = card[r];

        card[r] = card[i];

        card[i] = temp;

    }

    //    //For test

    //    for (int i = 0; i < 8; i++)

    //        cout << card[i] << " ";

}


void Game::swap() {
    if (p1.getBdName() == NULL || p2.getBdName() == NULL || p1.getBdFront() == NULL ||
p2.getBdFront() == NULL) return;
    Link *end1 = p1.endLst(1);
    Link *end2 = p2.endLst(1);
    Link *temp;
    temp->data = end1->data;
    end1->data = end2->data;
    end2->data = temp->data;

    Link *end3 = p1.endLst(2);
    Link *end4 = p2.endLst(2);
    temp->data = end3->data;
    end3->data = end4->data;
    end4->data = temp->data;
}

void Game::move(int p, int n) {
    if (p == 1)
        p1.setPosition(n);
    else if (p == 2)
        p2.setPosition(n);
    else return;
```

```
}

void Game::jump(int p, int pos) {
    if (p == 1) {
        p1.setJump(pos);
        p1.sptMoney(100);
    } else if (p == 2) {
        p2.setJump(pos);
        p2.sptMoney(100);
    } else return;
}

bool Game::compare() {
    if (p1.getBdFront() == NULL || p2.getBdFront() == NULL) return false;
    Link *end1 = p1.endLst(2);
    Link *end2 = p2.endLst(2);
    if (end1->data > end2->data) return true;
    else return false;
}

void Game::tripAI() {
    int pos;
    if (p2.getMoney() > 500) {
        if (rand() % 5 < 3) {
            do {
                pos = rand() % 24;
            } while (p1.search(1, pos) || pos == 12 || pos == 19 || pos == 22);
            cout << "\nComputer jumps to position #" << pos << endl;
            jump(2, pos);
        }
    }
}

void Game::jail(int p) {
    if (p == 1) {
        if (p1.getJTurn() == 0 && p1.getJail() == false) {
            p1.inJail();
            p1.jailTurn();
            //          p1.setJump(7);
        }
    } else if (p == 2) {
        if (p2.getJTurn() == 0 && p2.getJail() == false) {
            p2.inJail();
            p2.jailTurn();
        //          p2.setJump(7);
      }
   } else return;
}

bool Game::jailJudge(int n) {
   if (n == 1) {
      if (p1.getMoney() <= 100) return false;
      else return true;
   } else return true;
}

void Game::outJail(int p, int n) {
   if (p == 1) {
      if (n == 1) {
         cout << "You spent 100K to get out of jail." << endl;
         p1.sptMoney(100);
         p1.oJail();
      } else {
         char roll1, roll2;
         setDices();
         roll1 = getDice1();
```

```
            roll2 = getDice2();
            cout << "You choose to roll dices." << endl;
            cout << "Your result is: " << roll1 << " " << roll2 << endl << endl;
            if (roll1 == roll2) {
                cout << "Congratulation! You rolled double." << endl;
                p1.oJail();
            } else {
                cout << "Sorry, you did not get double in this turn." << endl;
                if (p1.getJTurn() == 3) {
                    cout << "You paid 50K to get out of jail." << endl;
                    p1.sptMoney(50);
                    p1.oJail();
                } else p1.jailTurn();
            }
        }
    } else if (p == 2) {
        if (p2.getMoney() > 400) {
            cout << "Computer spent 100K to get out of jail" << endl;
            p2.sptMoney(100);
            p2.oJail();
        } else {
            char roll1, roll2;
            setDices();
            roll1 = getDice1();
            roll2 = getDice2();
            cout << "Computer choose to roll dices." << endl;
            cout << "Computer's result is: " << roll1 << " " << roll2 << endl << endl;
            if (roll1 == roll2) {
                cout << "Computer rolled double and get out of jail." << endl;
                p2.oJail();
            } else {
                cout << "Computer did not get double." << endl;
                if (p2.getJTurn() == 3) {
                    cout << "computer paid 50K to get out of jail." << endl;
                    p2.sptMoney(50);
                    p2.oJail();
                } else p2.jailTurn();
            }
        }
    }
}

void Game::buyBrnd(int p, int pos) {
    if (p == 1) {
        p1.sptMoney(sell[pos]);

        p1.addBdName(pos);

        p1.addBdFront(coll[pos]);

    } else if (p == 2) {

        p2.sptMoney(sell[pos]);

        p2.addBdName(pos);

        p2.addBdFront(coll[pos]);

    } else return;

}


void Game::buyElec(int p) {

    if (p == 1) {

        p1.sptMoney(sell[9]);

        p1.addBdName(9);

        p1.addBdFront(coll[9]);

        elcNum--;

    } else if (p == 2) {

        p2.sptMoney(sell[9]);

        p2.addBdName(9);

        p2.addBdFront(coll[9]);

        elcNum--;

    } else return;
```

```
}

void Game::payRent(int p, int pos) {
   if (p == 1) {
      p1.sptMoney(sell[pos]);
      p2.takMoney(sell[pos]);
   } else if (p == 2) {
      p2.sptMoney(sell[pos]);
      p1.takMoney(sell[pos]);
   } else return;
}

void Game::remove(int p) {
   if (p == 1) {
      p1.pop(1);
      p1.pop(2);
   } else if (p == 2) {
      p2.pop(1);
      p2.pop(2);
   } else return;
}
```

```
int Game::getJail(int p) {
    if (p == 1) return p1.getJail();
    else if (p == 2) return p2.getJail();
    else return false;
}


void Game::title() {
    cout << "                              [][][][][]                              " << endl;
    cout << "                           [][][]    [][][]                           " << endl;
    cout << "[][][][][][][][][][][][][][][][][][][][][][]          [][][][][][][][][][][][][][][][][][][][][][]" <<
endl;
    cout << "[]                                                                  []" << endl;
    cout << "[]  []       []   [][][]   []     []   [][][]   [][][]    [][][]   []   []     []  []" << endl;
    cout << "[]  [][]     [][]  []     []  [][]   []  []     []  []   []  []     []  []   []     []  []" << endl;
    cout << "[]  []  []  []  []  []     []  []  []  []  []     []  []   []  []     []  []     []  []   []" << endl;
    cout << "[]  []    []    []  []     []  []   [][]  []     []  [][][]   []     []  []      []     []" << endl;
    cout << "[]  []       []   [][][]   []     []   [][][]   []        [][][]   [][][]  []       []" << endl;
    cout << "[]                                                                  []" << endl;
    cout << "[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]"
<< endl;
    cout << "\nPress enter to continue." << endl;
}
```

```
void Game::output() {

   cout << "
________________________________________________________________________________________________________
_________________________ " << endl;
   cout << "|   FREE    | GUITARHERO |   CHANCE   |   YAHOO!   |    Ford    |   CHANCE   |
ebay    |   GO TO    | " << "Player's Info:" << endl;
   cout << "|   PARKING  |    LIVE    |    CARD    |            |            |    CARD    |            |  JAIL!
| " << "------------------------" << endl;
   cout << "|            | 250K (150) |            | 250K (150) | 300K (150) |            | 300K (150) |
| " << p1.getName() << endl;
   cout << "|            |   (" << sale(13) << ")    |            |   (" << sale(15) << ")    |   (" << sale(16)
<< ")    |            |   (" << sale(18) << ")    |            | " << "Money: " << p1.getMoney() << "K" <<
endl;
   cout << "|  " << stopAt(12) << "  |  " << stopAt(13) << "  |  " << stopAt(14) << "  |  " <<
stopAt(15) << "  |  " << stopAt(16) << "  |  " << stopAt(17) << "  |  " << stopAt(18) << "  |
| " << "Brand Value: " << endl;
   cout << "|_____12_____|_____13_____|_____14_____|_____15_____|_____16_____|
_____17_____|_____18_____|_____19_____| ";
   p1.prntLst(2);
   cout << "|   Candy    |                                                          | UNIVERSAL  | " << "--
---------------------" << endl;
   cout << "|   Crush    |                                                          |            | " << p2.get-
Name() << endl;
   cout << "| 200K (100) |                                                          | 350K (200) | " <<
"Money: " << p2.getMoney() << "K" << endl;
   cout << "|    (" << sale(11) << ")     |                                                        |    (" <<
sale(20) << ")     | " << "Brand Value: " << endl;
   cout << "|  " << stopAt(11) << "  |                                                             |  " <<
stopAt(20) << "  | ";
   p2.prntLst(2);
   cout << "|_____11_____|                                                          |_____20_____|" <<
endl;
   cout << "|    LEVIS   |                                                          |    XBOX    |" << endl;
   cout << "|            |                                                          |            |" << endl;
   cout << "| 200K (100) |                                                          | 350K (200) |" <<
endl;
   cout << "|    (" << sale(10) << ")     |                                                        |    (" <<
sale(21) << ")     |" << endl;
   cout << "|  " << stopAt(10) << "  |                                                             |  " <<
stopAt(21) << "  |" << endl;
```

```
    cout << "|_____10_____|                                                  |_____21_____|" <<
endl;
    cout << "|  ELECTRIC  |                                                  |   TOWER    |" <<
endl;
    cout << "|  COMPANY   |                                                  |    TAX     |" <<
endl;
    cout << "| 150K(50," << elcNum << ") |                                                  |
|" << endl;
    cout << "|            |                                                  |            |" << endl;
    cout << "|  " << stopAt(9) << "  |                                                  |  " <<
stopAt(22) << "  |" << endl;
    cout << "|_____09_____|                                                  |_____22_____|" <<
endl;
    cout << "|    PUMA    |                                                  |   skype    |" << endl;
    cout << "|            |                                                  |            |" << endl;
    cout << "| 150K (100) |                                                  | 400K (200) |" <<
endl;
    cout << "|    (" << sale(8) << ")     |                                                  |    (" <<
sale(23) << ")     |" << endl;
    cout << "|  " << stopAt(8) << "  |                                                  |  " <<
stopAt(23) << "  |" << endl;
```

```
   cout << "|_____08_____|
______________________________________________________________________________|
_____23_____|" << endl;
   cout << "|  IN JAIL!  |  POLAROID  |   CHANCE    |    RAZOR    |   CHANCE    |TRANS-
FORMERS|   RIVAL     |    START    |" << endl;
   cout << "|__" << stopAt(19) << "__|            |    CARD     |             |    CARD     |             |  TOW-
ER     |            |" << endl;
   cout << "|  PASS BY    | 100K (50)  |            | 100K (50)  |            |  50K (50)  |    TAX      |
GO!     |" << endl;
   cout << "|            |    (" << sale(6) << ")     |            |    (" << sale(4) << ")     |            |    (" <<
sale(2) << ")     |            |            |" << endl;
   cout << "|  " << stopAt(7) << "  |  " << stopAt(6) << "  |  " << stopAt(5) << "  |  " << stopAt(4)
<< "  |  " << stopAt(3) << "  |  " << stopAt(2) << "  |  " << stopAt(1) << "  |  " << stopAt(0) << "
|" << endl;
   cout << "|_____07_____|_____06_____|_____05_____|______04____|______03____|
_____02_____|_____01_____|_____00_____|" << endl;

}

string Game::defineC(int n) {
   string defi;
   switch (n) {
    case 0:
        defi = "Speed ahead!\nMove forward to 5 space.";
        break;
    case 1:
        defi = "Casino night!\nBoth you and your opponent roll.\nHighest-roller collects 200K
from the bank.";
        break;
    case 2:
        defi = "Profits soar!\nAdvance to GO to collect your tower value.";
        break;
    case 3:
        defi = "Launch your website!\nSales skyrocket!\nCollect 300K from the bank.";
        break;
    case 4:
        defi = "Solar power bonus!\nTake a free Electric Company billboard.\nAdd it to your
tower.\nIf none are available, do nothing.";
        break;
    case 5:
        defi = "Insider trading fine!\nPay the Bank 200K.";
        break;
    case 6:
            defi = "Tallest tower bonus!\nCheck what the highest tower is currently worth.\nCollect
that amount from the bank.";

            break;
        default:
            defi = "Go To Jail!\nDo not collect cash for passing GO.";
    }
    return defi;
}

void Game::chanCrd(int p) {
    int n = rand() % 8;
    cout << "You pick Chance Card #" << n << endl << endl;
    cout << "------------------------------------------" << endl;
    cout << defineC(n) << endl;
    cout << "------------------------------------------" << endl << endl;
    switch (n) {
        case 0:
            if (p == 1) {
                cout << "You move forward 5 steps." << endl;
                p1.setPosition(5);
            } else {
```

```
            cout << "Computer moves forward 5 steps." << endl;
            p2.setPosition(5);
        }
        break;
    case 1:
        cout << "Both you and Computer roll." << endl << endl;
        char roll1, roll2;
        setDices();
        roll1 = getDice2();
        setDices();
        roll2 = getDice2();
        cout << "Your result is: " << roll1 << endl;
        cout << "Computer's result is: " << roll2 << endl << endl;
        if (roll1 > roll2) {
            cout << "You got higher number; Bank gives you 200K in reward." << endl;
            p1.takMoney(200);
        } else if (roll1 < roll2) {
            cout << "Computer got higher number; Computer collect 200K from bank." << endl;
            p2.takMoney(200);
        } else {
            cout << "Both you and Computer get same number; both of you can collect 200K." <<
endl;
```

```
        p1.takMoney(200);
        p2.takMoney(200);
    }
    break;
case 2:
    if (p == 1) {
        p1.setSum();
        cout << "You collect " << p1.getSum() << "K from Bank." << endl;
        p1.takMoney(p1.getSum());
    } else {
        p2.setSum();
        cout << "Computer collects " << p2.getSum() << "K from Bank." << endl;
        p2.takMoney(p2.getSum());
    }
    break;
case 3:
    if (p == 1) {
        cout << "You collect 300K from Bank." << endl;
        p1.takMoney(300);
    } else {
        cout << "Computer collects 300K from Bank." << endl;
        p2.takMoney(300);
```

```
  }
  break;
case 4:
  if (elcNum > 0) {
    if (p == 1) {
      cout << "You collect a free Electric Company billboard." << endl;
      p1.addBdName(9);
      p1.addBdFront(50);
    } else {
      cout << "Computer collects a free Electric Company billboard." << endl;
      p2.addBdName(9);
      p2.addBdFront(50);
    }
    elcNum--;
  } else {
    cout << "Sorry,there are no Electric Company billboards left." << endl;
  }
  break;
case 5:
  if (p == 1) {
    cout << "You pay Bank 200K." << endl;
    p1.sptMoney(200);
    } else {
        cout << "Computer pays Bank 200K." << endl;
        p2.sptMoney(200);
    }
    break;
case 6:
    int amount;
    p1.setSum();
    p2.setSum();
    if (p1.getSum() >= p2.getSum()) {
        amount = p1.getSum();
        cout << "You have a higher tower of " << amount << "K." << endl;
    } else {
        amount = p2.getSum();
        cout << "Computer has a higher tower of " << amount << "K." << endl;
    }
    if (p == 1) {
        cout << "You collect " << amount << "K from Bank." << endl;
        p1.takMoney(amount);
    } else {
        cout << "Computer collects " << amount << "K from Bank." << endl;
        p2.takMoney(amount);
```

```
            }
            break;
        default:
            if (p == 1) {
                cout << "You are sending to jail!" << endl;
                jail(1);
                p1.setJump(19);
            } else {
                cout << "Computer is sending to jail!" << endl;
                jail(2);
                p2.setJump(19);
            }
    }
}

int Game::getPos(int p) {
    if (p == 1) {
        return p1.getPostion();
    } else {
        return p2.getPostion();
    }
}
```

```
int Game::getMoney(int p) {
   if (p == 1) {
      return p1.getMoney();
   } else {
      return p2.getMoney();
   }
}

string Game::getName(int p) {
   if (p == 1) {
      return p1.getName();
   } else {
      return p2.getName();
   }
}

void Game::rules() {
   cout << "****************** Monopoly: Empire Rules
********************************" << endl;
   cout << "I. Basic rules" << endl;
   cout << "   A. Each player has 1000K at beginning and start at position #0." << endl;
   cout << "   B. In each turn, player roll 2 dices, and move corresponding steps." << endl;
   cout << "   C. When stop at a position, follow the instruction of that position." << endl;
   cout << "   D. [1] represent player, [2] represent computer." << endl;
   cout << endl;
   cout << "II. Game Board" << endl;
   cout << "   A. Total 24 positions (0~24) of 7 types." << endl;
   cout << "   B. Start" << endl;
   cout << "       1. Every player start from here." << endl;
   cout << "       2. When pass start point, collect as much money as your total collecting value."
<< endl;
   cout << "   C. Billboard" << endl;
   cout << "       1. Containing name, price, collecting value, and condition(0,1 or 2)." << endl;
   cout << "       2. The first two lines are the name of the brand." << endl;
   cout << "       3. The third line is price (collecting value), eg.100K (50)." << endl;
   cout << "           a. Price ≠ collecting value." << endl;
   cout << "           b. Price is how much cost you to buy the brand." << endl;
   cout << "           c. Collecting value is used for win the game." << endl;
   cout << "       4. The fourth line is (condition), eg. (1)." << endl;
   cout << "            (0 = no owner; 1 = player own; 2 = computer own)" << endl;
   cout << "       5. The fifth line shows player's position, eg.[1] [2]." << endl;
   cout << "       6. The bottom line has the position # of that box." << endl;
   cout << "       7. Exception: Electric Company Billboard. (See IV. Billboard-E)" << endl;
```

```
    cout << "   D. Chance Card" << endl;
    cout << "       1. Randomly pick a chance card when stop at here." << endl;
    cout << "       2. Total 8 kinds of chance cards." << endl;
    cout << "       3. Chance card may not be good for player who pick it." << endl;
    cout << "   E. Jail" << endl;
    cout << "       1. To get out of jail, there are two choice." << endl;
    cout << "           a. During next turn, pay 100K and get out at once." << endl;
    cout << "           b. Roll a double up to 3 turns, if not, pay 50K and get out at 3rd turn." <<
endl;
    cout << "       2. Position #19 is go to jail, player will be move to jail on Position #7." << endl;
    cout << "       3. Simply stop at position #7 is pass by, is not in jail." << endl;
    cout << "   F. Take a trip" << endl;
    cout << "       1. You can spend 100K to move to anywhere on board, or simply do nothing."
<< endl;
    cout << "       2. If you move to a new position, you have to follow the instruction on that posi-
tion." << endl;
    cout << "   E. Tower Tax" << endl;
    cout << "       1. Tower Tax: return your topmost billboard to board." << endl;
    cout << "       2. Rival Tower Tax: return your opponent's topmost billboard to board." << endl;
    cout << endl;
    cout << "III. Dices" << endl;
    cout << "   A. One normal dice, one special dice with a 'swap' face." << endl;
    cout << "   B. When 'swap' face is up, you have a chance of sneaky swap." << endl;
    cout << "       (sneaky swap: swap your topmost billboard with your opponent's)" << endl;
    cout << "   C. When you roll a double, you can have another turn until you don't have a dou-
ble." << endl;
    cout << "       (double: two dices have same number)" << endl;
    cout << endl;
    cout << "IV. Billboard" << endl;
    cout << "   A. 14 billboards total; different price with different collecting value; choose smart-
ly!" << endl;
    cout << "   B. You can buy billboards when you stop at the corresponding position if the con-
dition is no owner." << endl;
    cout << "   C. You have to pay rent fee if you stop at a position whose billboard is owner by
your opponents." << endl;
    cout << "   D. Nothing will happen if you stop at the position that is owned by yourself." <<
endl;
    cout << "   E. One exception is Electric Company Billboard at position #9." << endl;
    cout << "       1. Has 4 in total" << endl;
    cout << "       2. Doesn't own by anyone." << endl;
    cout << endl;
    cout << "V. Chance Card" << endl;
    cout << "   A. Suggestion: Better read card function before playing game." << endl;
    cout << "   B. Explanation of each card:" << endl;
```

```
   cout << "       1.Speed ahead!\nMove forward to 5 space." << endl;
   cout << "       2.Casino night!\nBoth you and your opponent roll.\nHighest-roller collects 200K
from the bank." << endl;
   cout << "       3.Profits soar!\nAdvance to GO to collect your tower value." << endl;
   cout << "       4.Launch your website!\nSales skyrocket!\nCollect 300K from the bank." <<
endl;
   cout << "       5.Solar power bonus!\nTake a free Electric Company billboard.\nAdd it to your
tower.\nIf none are available, do nothing." << endl;
   cout << "       6.Insider trading fine!\nPay the Bank 200K." << endl;
   cout << "       7.Tallest tower bonus!\nCheck what the highest tower is currently worth.\nCol-
lect that amount from the bank." << endl;
   cout << "       8.Go To Jail!\nDo not collect cash for passing GO." << endl;
   cout << endl;
   cout << "VI. Win" << endl;
   cout << "   A. Whoever first get 800 collecting value wins." << endl;
   cout << "   B. Whoever first run out of money loses." << endl;
   cout << endl;
   cout << "VII. Bonus" << endl;
   cout << "   The bonus area will be open for player each time when player win a game." <<
endl;
   cout << "VIII. Tips For Brand Buying" << endl;
```

```
    cout << "   A. The unit value for different brand is different. Be wise to choose which brand to
buy!" << endl;
    cout << "   B. The brand unit value sorting (large -> small):" << endl << endl;
    valueSort();
    cout << endl;
}
```

BNTnode.h

```
/*
 * File:   BNTnode.h
 * Modified:  from http://www.sanfoundry.com/cpp-program-implement-avl-trees/
 * Created on May 23, 2017, 9:14 PM
 * Purpose:  A binary tree node
 */

#ifndef BNTNODE_H
#defineBNTNODE_H


struct BNTnode{
    int data;
    struct BNTnode *left;
    struct BNTnode *right;
};

#endif  /* BNTNODE_H */
```

AVLTree.h

```
/*
 * File:   BNTnode.h
 * Modified:  from http://www.sanfoundry.com/cpp-program-implement-AVL-trees/
 * Created on May 23, 2017, 9:14 PM
 * Purpose:  An Binary Tree using an AVL balancing technique
 */

#ifndef AVLTREE_H
#define AVLTREE_H

#include <iostream>
using namespace std;

#include "BNTnode.h"

class AVLTree {
public:
   BNTnode *root; //Root node

   int height(BNTnode *); //Tree height

   int diff(BNTnode *); //Difference of right/left subtrees

   BNTnode *rr_rotation(BNTnode *); //Right-Right rotation

   BNTnode *ll_rotation(BNTnode *); //Left-Left   rotation

   BNTnode *lr_rotation(BNTnode *); //Left-Right  rotation

   BNTnode *rl_rotation(BNTnode *); //Right-Left  rotation

   BNTnode* balance(BNTnode *); //Balance subtrees with diff > 1

   BNTnode* insert(BNTnode *, int); //Insert and balance the tree

   void display(BNTnode *, int); //Funky display root left to right

   void inorder(BNTnode *); //In order display

   void preorder(BNTnode *); //Pre order display

   void postorder(BNTnode *); //Post order display


   AVLTree() {

      root = NULL;

   } //Constructor

};


//
******************************************************************************

//                    Height of AVL Sub Trees
```

```
//
*******************************************************************************

int AVLTree::height(BNTnode *temp) {
   int h = 0;
   if (temp != NULL) {
      int l_height = height(temp->left);
      int r_height = height(temp->right);
      int max_height = max(l_height, r_height);
      h = max_height + 1;
   }
   return h;
}

//
*******************************************************************************
//                    Height Difference of AVL Sub Trees
//
*******************************************************************************

int AVLTree::diff(BNTnode *temp) {
   int l_height = height(temp->left);
   int r_height = height(temp->right);

   int b_factor = l_height - r_height;

   return b_factor;

}


//
****************************************************************************

//                    Right-Right Rotations of Sub Trees

//
****************************************************************************


BNTnode *AVLTree::rr_rotation(BNTnode *parent) {

   BNTnode *temp;

   temp = parent->right;

   parent->right = temp->left;

   temp->left = parent;

   return temp;

}


//
****************************************************************************

//                    Left-Left Rotations of Sub Trees
```

```
//
******************************************************************************

BNTnode *AVLTree::ll_rotation(BNTnode *parent) {
   BNTnode *temp;
   temp = parent->left;
   parent->left = temp->right;
   temp->right = parent;
   return temp;
}

//
******************************************************************************
//                  Left-Right Rotations of Sub Trees
//
******************************************************************************

BNTnode *AVLTree::lr_rotation(BNTnode *parent) {
   BNTnode *temp;
   temp = parent->left;
   parent->left = rr_rotation(temp);
   return ll_rotation(parent);
```

```
}

//
******************************************************************************
//                  Right-Left Rotations of Sub Trees
//
******************************************************************************

BNTnode *AVLTree::rl_rotation(BNTnode *parent) {
   BNTnode *temp;
   temp = parent->right;
   parent->right = ll_rotation(temp);
   return rr_rotation(parent);
}

//
******************************************************************************
//                    Balancing of Sub Trees
//
******************************************************************************

BNTnode *AVLTree::balance(BNTnode *temp) {
   int bal_factor = diff(temp);

   if (bal_factor > 1) {

       if (diff(temp->left) > 0)

           temp = ll_rotation(temp);

       else

           temp = lr_rotation(temp);

   } else if (bal_factor < -1) {

       if (diff(temp->right) > 0)

           temp = rl_rotation(temp);

       else

           temp = rr_rotation(temp);

   }

   return temp;

}


//
********************************************************************************
//                  Insert the Data into the Sub Trees
//
********************************************************************************


BNTnode *AVLTree::insert(BNTnode *root, int value) {
   if (root == NULL) {
       root = new BNTnode;
       root->data = value;
       root->left = NULL;
       root->right = NULL;
       return root;
   } else if (value < root->data) {
       root->left = insert(root->left, value);
       root = balance(root);
   } else if (value >= root->data) {
       root->right = insert(root->right, value);
       root = balance(root);
   }
   return root;
}


//
********************************************************************************
//                      Display all Sub Trees
//
********************************************************************************
```

```
void AVLTree::display(BNTnode *ptr, int level) {
   int i;
   if (ptr != NULL) {
      display(ptr->right, level + 1);
      printf("\n");
      if (ptr == root)
         cout << "R -> ";
      for (i = 0; i < level && ptr != root; i++)
         cout << "     ";
      cout << ptr->data;
      display(ptr->left, level + 1);
   }
}

//
****************************************************************************
//                  In-order Output of Tree
//
****************************************************************************

void AVLTree::inorder(BNTnode *tree) {
   if (tree == NULL)
```

```
        return;
    inorder(tree->left);
    cout << tree->data << "  ";
    inorder(tree->right);
}

//
*******************************************************************************
//                     Pre-order Output of Tree
//
*******************************************************************************

void AVLTree::preorder(BNTnode *tree) {
    if (tree == NULL)
        return;
    cout << tree->data << "  ";
    preorder(tree->left);
    preorder(tree->right);
}

//
*******************************************************************************
```

```
//                    Post-order Output of Tree
//
******************************************************************************

void AVLTree::postorder(BNTnode *tree) {
   if (tree == NULL)
      return;
   postorder(tree ->left);
   postorder(tree ->right);
   cout << tree->data << "  ";
}

#endif /* AVLTREE_H */
```

GeneralHashFunctions.h

```
/*
 **************************************************************************
 *                                                   *
 *         General Purpose Hash Function Algorithms Library            *
 *                                                   *
 * Author: Arash Partow - 2002                            *
 * URL: http://www.partow.net                             *
 * URL: http://www.partow.net/programming/hashfunctions/index.html        *
 *                                                                        *
 * Copyright notice:                                                      *
 * Free use of the General Purpose Hash Function Algorithms Library is    *
 * permitted under the guidelines and in accordance with the MIT License. *
 * http://www.opensource.org/licenses/MIT                                 *
 *                                                                        *
 **************************************************************************
*/


#ifndef INCLUDE_GENERALHASHFUNCTION_CPP_H
#define INCLUDE_GENERALHASHFUNCTION_CPP_H


#include <string>


typedef unsigned int (*HashFunction)(const std::string&);


unsigned int RSHash  (const std::string& str);
unsigned int JSHash  (const std::string& str);

unsigned int PJWHash (const std::string& str);

unsigned int ELFHash (const std::string& str);

unsigned int BKDRHash(const std::string& str);

unsigned int SDBMHash(const std::string& str);

unsigned int DJBHash (const std::string& str);

unsigned int DEKHash (const std::string& str);

unsigned int BPHash  (const std::string& str);

unsigned int FNVHash (const std::string& str);

unsigned int APHash  (const std::string& str);



#endif
```

GeneralHashFunctions.cpp

```
/*
 **************************************************************************
 *                                                        *
 *          General Purpose Hash Function Algorithms Library              *
 *                                                        *
 * Author: Arash Partow - 2002                                    *
 * URL: http://www.partow.net                                      *
 * URL: http://www.partow.net/programming/hashfunctions/index.html        *
 *                                                                        *
 * Copyright notice:                                                      *
 * Free use of the General Purpose Hash Function Algorithms Library is    *
 * permitted under the guidelines and in accordance with the MIT License. *
 * http://www.opensource.org/licenses/MIT                                 *
 *                                                                        *
 **************************************************************************
*/


#include "GeneralHashFunctions.h"


unsigned int RSHash(const std::string& str)
{
   unsigned int b    = 378551;
   unsigned int a    = 63689;
   unsigned int hash = 0;

   for(std::size_t i = 0; i < str.length(); i++)
   {
      hash = hash * a + str[i];
      a    = a * b;
   }

   return hash;
}
/* End Of RS Hash Function */



unsigned int JSHash(const std::string& str)
{
   unsigned int hash = 1315423911;

   for(std::size_t i = 0; i < str.length(); i++)
   {
      hash ^= ((hash << 5) + str[i] + (hash >> 2));
   }

   return hash;
}
/* End Of JS Hash Function */
```

```
unsigned int PJWHash(const std::string& str)
{
  unsigned int BitsInUnsignedInt = (unsigned int)(sizeof(unsigned int) * 8);
  unsigned int ThreeQuarters     = (unsigned int)((BitsInUnsignedInt  * 3) / 4);
  unsigned int OneEighth         = (unsigned int)(BitsInUnsignedInt / 8);
  unsigned int HighBits          = (unsigned int)(0xFFFFFFFF) << (BitsInUnsignedInt - One-
Eighth);
  unsigned int hash              = 0;
  unsigned int test              = 0;

  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash = (hash << OneEighth) + str[i];

    if((test = hash & HighBits)  != 0)
    {
      hash = (( hash ^ (test >> ThreeQuarters)) & (~HighBits));
    }
  }

  return hash;
}

/* End Of  P. J. Weinberger Hash Function */



unsigned int ELFHash(const std::string& str)

{

  unsigned int hash = 0;

  unsigned int x    = 0;


  for(std::size_t i = 0; i < str.length(); i++)

  {

    hash = (hash << 4) + str[i];

    if((x = hash & 0xF0000000L) != 0)

    {

      hash ^= (x >> 24);

    }

    hash &= ~x;

  }


  return hash;

}

/* End Of ELF Hash Function */
```

```
unsigned int BKDRHash(const std::string& str)
{
  unsigned int seed = 131; // 31 131 1313 13131 131313 etc..
  unsigned int hash = 0;

  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash = (hash * seed) + str[i];
  }

  return hash;
}
/* End Of BKDR Hash Function */


unsigned int SDBMHash(const std::string& str)
{
  unsigned int hash = 0;

  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash = str[i] + (hash << 6) + (hash << 16) - hash;
  }

  return hash;
}
/* End Of SDBM Hash Function */


unsigned int DJBHash(const std::string& str)
{
  unsigned int hash = 5381;

  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash = ((hash << 5) + hash) + str[i];
  }

  return hash;
}
/* End Of DJB Hash Function */
```

```
unsigned int DEKHash(const std::string& str)
{
  unsigned int hash = static_cast<unsigned int>(str.length());

  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash = ((hash << 5) ^ (hash >> 27)) ^ str[i];
  }

  return hash;
}
/* End Of DEK Hash Function */


unsigned int BPHash(const std::string& str)
{
  unsigned int hash = 0;
  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash = hash << 7 ^ str[i];
  }
  return hash;
}
/* End Of BP Hash Function */



unsigned int FNVHash(const std::string& str)
{
  const unsigned int fnv_prime = 0x811C9DC5;
  unsigned int hash = 0;
  for(std::size_t i = 0; i < str.length(); i++)
  {
    hash *= fnv_prime;
    hash ^= str[i];
  }

  return hash;
}
/* End Of FNV Hash Function */



unsigned int APHash(const std::string& str)
```

```
{
   unsigned int hash = 0xAAAAAAAA;


   for(std::size_t i = 0; i < str.length(); i++)
   {
      hash ^= ((i & 1) == 0) ? (  (hash <<  7) ^ str[i] * (hash >> 3)) :
                               (~((hash << 11) + (str[i] ^ (hash >> 5))));
   }


   return hash;
}
/* End Of AP Hash Function */
```