Assignment #2

Computer Science I – COP 3502

Objectives

1. To learn how to create and implement linked lists

Problem: Monopoly Tracker (monopoly.c)

Monopoly is a popular board game in which players attempt to amass wealth and property and force any opposing players to bankrupt. Properties can be bought from the bank and built upon to increase their value.

Our program will create a tracker for players of the Monopoly game. Up to six players may play Monopoly, so the first thing we should do is ask how many people are playing the game. Our tracker will keep a list of purchased properties, who owns the property, and their number of improvements (built houses and hotels).

The purchased properties will be stored in a linked list. Each node of the list should have the following structure.

```
struct property {
        char name[30];
        int user;
        int num_houses;
        int num_hotels;
        struct property * next;
}
```

For each user's turn, the following menu should be printed. A detailed description of each function is given below.

1. Buy Property
2. Improve Property
3. Sell Property
4. View Properties
5. End Turn
6. End Game

The program should continue to print this menu for a player until either End Turn or End Game is selected. If End Turn is selected, the program should print this menu for the next player. This process should continue for all players – returning to player 1 after the last player has gone – until a player selects End Game.

1) Buy Property (function required)
If the user selects option one, the program should ask the user for the name of the property they wish to buy. The program should compare the player's answer to the list of already purchased properties.

If the property already belongs to someone, the program should print one of the following statements:

Player X owns that property.
You already own that property!

If the property does not belong to anyone, the program should add the new property to the purchased list with current players number, 0 houses, and 0 hotels.

2) Improve Property (function required)
If the user selects option two, the program should ask the user for the name of the property they wish to improve. The program should compare the player's answer to the list of already purchased properties to ensure the user already owns that property.

If the user does not own the property specified (either because another player owns or because the property has not been purchased yet), print the following:

You do not own that property.

If the user does own the property, ask if the user wishes to build houses or hotels. Only one type can be purchased at a time. Based on their selection, ask how many they wish to purchase and increase either num_houses or num_hotels accordingly.

3) Sell Property (function required)
If the user selects option three, the program should ask the user for the name of the property they wish to sell. The program should compare the player's answer to the list of already purchased properties to ensure the user already owns that property.

If the user does not own the property specified (either because another player owns or because the property has not been purchased yet), print the following:

You do not own that property.

If the user does own the property, calculate the final value of the property. For simplicity, each property is worth 100 game dollars. Each hotel added to the property is worth 50 game dollars, and each house added to the property is worth 25 game dollars. Print the following:

You sold <Property> for $X.XX!

4) View Properties
If the user selects option four, the program should print the list of purchased properties using the following format for each property. The properties may be in any order.

<Name of Property>, owned by <user number>

5) End Turn
If the user selects option five, the program should move on the to the next player. For example, if there are four players, the turns would be player 1, player 2, player 3, player 4, player 1, player 2, etc.

6) End Game
If the user selects option six, the program should print a final listing of all the purchased properties (see option four) and quit.

Program Specifications

1. The number of users playing will be a positive integer between 1 and 6 (inclusive)
2. Menu input will be a positive integer
3. Users will enter property names without spaces in them
    a. For example, a player may enter Baltic_Avenue instead of Baltic Avenue
4. You must write separate functions to complete the following tasks. The functions' parameters and prototypes will be up to you. You may also add additional functions as you see fit.
    a. Buy Property
    b. Improve Property
    c. Sell Property
    d. Add property to list
    e. Remove property from list
    f. Print list
5. You **must** use the structure listed in the problem description to form a linked list. Programs that do not use linked lists will not be graded.

Sample Run
Below is a sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above.

In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Welcome to the Monopoly Property Tracker!

How many people are playing?
3

Player 1, what would you like to do?
1 - Buy Property
2 - Improve Property
3 - Sell Property
4 - View Properties
5 - End Turn
6 - End Game
1

What is the name of the property you wish to buy?
Baltic_Avenue

Player 1, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

2

Which property would you like to improve?
Baltic_Avenue

Do you wish to build houses(1) or hotels(2)?
2

How many hotels do you want to build?
1

Player 1, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

4

Baltic_Avenue, owned by Player 1

Player 1, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

3

Which property would you like to sell?
St_James_Place

You do not own that property.

Player 1, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

3

Which property would you like to sell?
Baltic_Avenue

You sold Baltic_Avenue for $150.00!

Player 1, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

5

Player 2, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

1

What is the name of the property you wish to buy?
Marvin_Gardens

Player 2, what would you like to do?
- **1 - Buy Property**
- **2 - Improve Property**
- **3 - Sell Property**
- **4 - View Properties**
- **5 - End Turn**
- **6 - End Game**

6

Marvin_Gardens, owned by Player 2

Deliverables

One source files – *monopoly.c* – is to be submitted over WebCourses.

Restrictions

Your program must be able to be compiled and executed in a standard C Development Environment. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.