Homework 4: Analyzing Monopoly

**Due Sunday, October 22nd by 11:59pm**

You are going to do a simple analysis of Monopoly to see what spaces get landed on most often. To do this, you’ll need to simulate the roll of the dice and keep track of the number of times you land on each space. Finally, you’ll need to compute values to see which spaces are most popular.

# Task Overview

This is a brief overview of the tasks you must complete for this assignment. Specifics are given in the corresponding sections later in this document. You should ***read the entire document*** before starting.

* Create a **Dice class**, which can roll one or more dice and return the total
* Create a **Monopoly class**, that can simulate a character going around a Monopoly board
  + **Analyze** which spaces on a Monopoly board get landed on the most
  + **Print** the results of your analysis

# Activity 1: Rolling Dice

Create a new class called Dice in your project. This class will be in charge of simulating the roll of one or more six sided dice. To do this, you’ll need to use C#’s built-in random number generator class (called Random).

## Fields

Your Dice class will need one private field: a variable to hold a Random object.

## Constructor

The Dice class only needs a default constructor. It should simply create a random object and store it in the class’s field.

## int RollDie()

This method should simulate the roll of a single six-sided die. It should return a random integer between 1 and 6 inclusive. Remember that the Random class’s “Next” method’s maximum value is *exclusive*, so you’ll need to compensate for that.

## int RollDice(int amount)

This method should simulate the roll of multiple six sided dice. Make use of the RollDie() method here, since you’ve already done that work. Simply call it the appropriate number of times, totaling up the results, and then returning that overall value.

**Note:** Making the random number generator produce a value from 2 to 12 will not act the same as rolling two separate dice. Think about the kinds of values you might get from rolling two dice – and why the number seven is considered “lucky”.

## Main Method: Testing your Dice

Once you have finished the two methods, create a Dice object in your Main method and do 100 calls to each of your two Dice class methods, printing out the results to make sure they generate numbers in the appropriate range.

# Activity 2: Looking at Monopoly

The game Monopoly has been around since the 1930’s. See the included Monopoly Board document for an example of a standard Monopoly board layout.

In many cases, game designers spend time balancing a game. They look for things like dominant strategies (a strategy almost guaranteed to win a game if employed correctly), unbalanced units, or specific places that seem to provide an advantage or disadvantage. A lot of this balancing comes from playtesting, but there are some cases when a mathematical analysis similar to the one we are about to do would prove useful.

It’s well known that there are certain spots in Monopoly that give the player an advantage, since they are landed on more often. By getting those locations and building them up, a player can give themselves a boost in collecting income.

This activity will be part of the same project as activity 1.

## Monopoly Class

Create a class named Monopoly, which will allow you to simulate rolling dice and moving around a monopoly board.

## Fields

The class will need several fields.

The first is a *Dice* object field (this will be an object of the class you made in activity 1).

The second field is a *String array* to hold the names of all the spaces. Rather than having to find and type all of the space names by hand, refer to the *MonopolyBoard.docx* file that came with the homework. The second page has code for a *literal array*, which you can paste into this class.

A literal array is an array in which all of the values are hardcoded into the program.

Set up two *constant* integer fields for the class. One will hold the *number of players* in the study. Set this to 100,000. The other should hold the number of times each player goes *around the board*. Set this to 25.

Remember that constant variables are created using the “const” keyword before the data type.

## Constructor

Create a default constructor for the class. It should initialize the Dice object.

## double[] Analyze()

Create a method called “Analyze”. This method will return an array of doubles representing the percent of times each space is visited. (You’ll calculate this as you go along.) It will not need any parameters. The Analyze method should do the following:

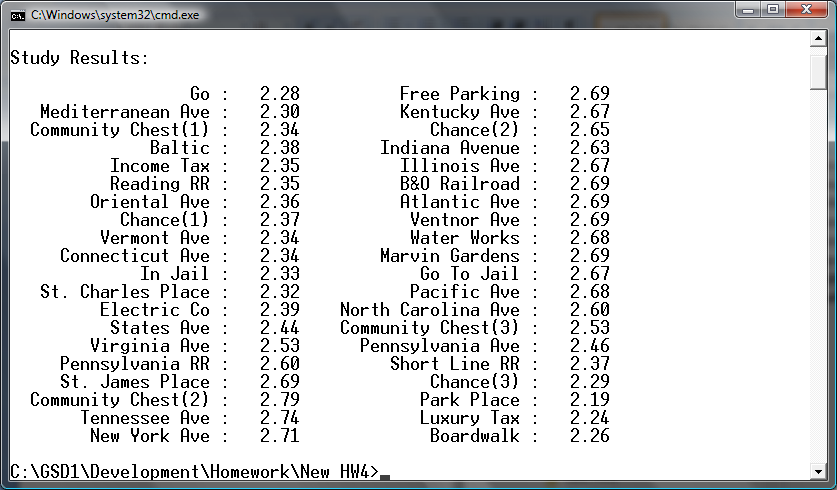
* You’ll need an integer array to hold the number of times each space on the board is visited. Call it ***visits***. It should have the same number of elements as the *spaces* array (the one with all the space names). Create the *visits* array inside this method.
* Create a loop that will be executed once for each player (remember you have a constant for the number of players). This will be the outer loop for the following steps.
* You need to track the current player’s location. Create an int variable to store the player’s current position. Assume they start at “Go” (position 0).
* You also need to track the number of times they have gone around the board. Start this variable at 1. This will be the loop control variable for the next step.
* Create an inner loop that will execute *until* the player has gone around the board the right number of times (another one of your constants). You do NOT know how many iterations this will take, since it’s based on random dice rolls. You’ll want to use a “while” loop here, not a “for” loop!
* Simulate a player’s turn by rolling two six sided dice. This is where you need to use the Dice object you created before. Roll two dice and move the player forward by updating their position variable. Mark the fact that they visited the appropriate space in the ***visits*** array. There are two special cases that need to be addressed:
  + If they pass “Go”, it means they’ve gone all the way around the board. Increment the “times around the board” count and make sure their position doesn’t go past the end of the array (If there are 40 spaces and they’ve gone to space 42, they’ve gone around once and should now be at space 2).
  + If they land on “Go to Jail”, increment the times around the board count because they start a new time around the board on their way to jail. Only mark the “Go to Jail” spot as the one visited. Do not mark the “Jail” spot as visited in this case.
* Once all loops have completed, you need to calculate the percentage of time the players landed on each space. Start by creating an array of doubles to hold the percentages. There needs to be one element in the array per space on the board.
* Get the total count of all the moves by adding up the counts stored in the “visits” array. (Hint: This involves another loop.)
* Calculate the percentage of all visits for each space and place the values in the percentages array created above. (Hint: Yet another loop).

Percent of visits to a space = (visits to a single space / total visits) \* 100

* Now that you have all of the percentages calculated, return the entire percentages array.

## void PrintResults(double[] visitPercentages)

Create another method called “PrintResults”. This method returns nothing. It accepts an array with the percentages of each visit and prints each to the console window. You can print out each space on a separate line, or attempt to organize it like the sample run below:



Remember: To format your numbers like this, you can use the format specifier F2 (or P2 for percents):

double num = 1234.5678;

Console.WriteLine("{0:F2}", num); // Rounds & prints: 1234.57

If you’d like your words to line up nicely like in the example, look up the PadLeft() method of Strings in C#. It will add an amount of extra characters (such as spaces, or whatever you specify) to the left side of a string to ensure the string ends up being the specified length.

If you want to truly match the output above, you’ll need to print two different pieces of data per line (although that isn’t required for this homework).

## Main Method

Now do the following in the Main method after your test code from activity 1:

* Create a Monopoly object.
* Call the Analyze method and save its results in a variable.
* Call the PrintResults method and pass in the results from the Analyze method.

Standard (American Edition) Monopoly

For your reference – Monopoly game board layout as of September 2008

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Free Parking](http://en.wikipedia.org/wiki/Free_Parking) | Kentucky Avenue $220 | [Chance](http://en.wikipedia.org/wiki/Chance_and_Community_Chest_cards#Chance)  **?** | Indiana Avenue $220 | Illinois Avenue $240 | [B&O Railroad](http://en.wikipedia.org/wiki/Baltimore_and_Ohio_Railroad) $200 | Atlantic Avenue $260 | Ventnor Avenue $260 | [Water Works](http://en.wikipedia.org/wiki/Water_supply) $150 | [Marvin Gardens](http://en.wikipedia.org/wiki/Marven_Gardens) $280 | Go To Jail |
| New York Avenue $200 | |  | | --- | | MONOPOLY | | | | | | | | | | Pacific Avenue $300 |
| Tennessee Avenue $180 | North Carolina Avenue $300 |
| [Community Chest](http://en.wikipedia.org/wiki/Chance_and_Community_Chest_cards#Community_Chest) | [Community Chest](http://en.wikipedia.org/wiki/Chance_and_Community_Chest_cards#Community_Chest) |
| St. James Place $180 | Pennsylvania Avenue $320 |
| [Pennsylvania Railroad](http://en.wikipedia.org/wiki/Pennsylvania_Railroad) $200 | [Short Line](http://en.wikipedia.org/wiki/Shore_Fast_Line) $200 |
| Virginia Avenue $160 | [Chance](http://en.wikipedia.org/wiki/Chance_and_Community_Chest_cards#Chance)  **?** |
| States Avenue $140 | Park Place $350 |
| [Electric Company](http://en.wikipedia.org/wiki/Electrical_power_industry) $150 | [Luxury Tax](http://en.wikipedia.org/wiki/Luxury_tax) (pay $100) |
| St. Charles Place $140 | Boardwalk $400 |
| In Jail/Just Visiting | Connecticut Avenue $120 | Vermont Avenue $100 | [Chance](http://en.wikipedia.org/wiki/Chance_and_Community_Chest_cards#Chance)  **?** | Oriental Avenue $100 | [Reading Railroad](http://en.wikipedia.org/wiki/Reading_Company) $200 | [Income Tax](http://en.wikipedia.org/wiki/Income_tax) (pay $200) | Baltic Avenue $60 | [Community Chest](http://en.wikipedia.org/wiki/Chance_and_Community_Chest_cards#Community_Chest) | Mediter-ranean Avenue $60 | **Go** Collect $200 salary as you pass [http://upload.wikimedia.org/wikipedia/commons/thumb/9/96/Monopoly_Go_Arrow.png/44px-Monopoly_Go_Arrow.png](http://en.wikipedia.org/wiki/File:Monopoly_Go_Arrow.png) |

<http://en.wikipedia.org/wiki/Monopoly_(game)>

Detailed probability info: <http://www.tkcs-collins.com/truman/monopoly/monopoly.shtml>

See the next page for a C# array of the Monopoly property names.

C# Array of Monopoly Property Names

Feel free to copy/paste the following into your Monopoly class for this homework assignment.

private String[] propertyNames = {

"Go",

"Mediterranean Avenue",

"Community Chest (1)",

"Baltic Avenue",

"Income Tax",

"Reading Railroad",

"Oriental Avenue",

"Chance (1)",

"Vermont Avenue",

"Connecticut Avenue",

"Jail",

"St. Charles Place",

"Electric Company",

"States Avenue",

"Virginia Avenue",

"Pennsylvania Railroad",

"St. James Place",

"Community Chest (2)",

"Tennessee Avenue",

"New York Avenue",

"Free Parking",

"Kentucky Avenue",

"Chance (2)",

"Indiana Avenue",

"Illinois Avenue",

"B&O Railroad",

"Atlantic Avenue",

"Ventnor Avenue",

"Water Works",

"Marvin Gardens",

"Go To Jail",

"Pacific Avenue",

"North Carolina Avenue",

"Community Chest (3)",

"Pennsylvania Avenue",

"Short Line Railroad",

"Chance (3)",

"Park Place",

"Luxury Tax",

"Boardwalk"

};