Lab 6

(Due Sunday, May 22) You may choose to work with a partner, or you may choose to work alone on this lab.

Note: This lab is worth 35 points. You may choose to work on 1, 2, or 3 of the problems, based on how many points you think you need. Equally, if you have been doing extra credit throughout the semester, you may have enough lab points that you may choose to forgo doing this lab altogether.

That said, problems 1 and 2 will be good practice working with collections.

Problem 1: (15 pts)

Elimination game. This is based loosely on "Eenie meanie miney moe" , where one would go around a circle counting until one got to "it" and "it" was eliminated from the circle. When I count the counts in this game I get 31.

For this game, you will create a class. The class will have 2 fields – one is an ArrayList and one is a LinkedList. For our purposes of testing the lists will both hold Integers.

In the constructor, initialize each list to be sequential integers up to a large number (I picked 5000).

The class will also have a method that takes as input a List. You do not need to specify whether it's an ArrayList or a Linked List. You only need to specify that the input parameter is of type List<Integer>. In this function the eliminations will take place. While the list still contains an element (its size is greater than 0), eliminate every 31st elements. So, for instance, you will remove the 31st element, then the 62nd element, then the 93rd element, etc. When you get to the end of the list, you should loop back to the beginning. So, for instance, if the size is 5000 when you get to 4991, the next one to be eliminated would be 5022. With only 5000 elements in the array in the first round, you'd then eliminate the 22 element in the array and continue from there. Note that you constantly have to check the size because the size is constantly changing. You should print out each one that's eliminated. When the list is empty, return from this function.

You will need one more method in this class. This method is going to time how long it takes to eliminate values from each of the two lists. To do this, you'll use a Date object.

```
Date date = new Date();
```

Once you've created a date object, you will want to get the current time in milliseconds:

```
long x = date.getTime();
```

You will then call the method that plays the game with your ArrayList field, and, when the method is done running, call the getTime() method again with another variable. The difference between the first getTime() and the second getTime() is the amount of time in milliseconds that it took to remove all the values from the ArrayList.

Next, within the same function, repeat the same process, calling the method with your LinkedList field.

Compare the two times. Which took longer, eliminating values from the ArrayList or the LinkedList? Do you understand why?

Problem 2: HashMaps (20 pts)

One of the things we're trying to teach the computer is to recognize different "voices" and in this case, I mean the patterns with which different people use words. For this example, we'll be emulating Charles Dickins' voice, although you could easily attempt to emulate Poe, or Shakespeare, and, with a smarter word predictor, we could use this on your phone to better predict what word you'd want to text next after texting a particular word.

More specifically, for this assignment, you will be reading in from Great Expectations. Each word read in will be a key, and the values associated with each key will be a set of Strings, or a set of words that follow a word. So, for instance, if you have the word "I" as your key, the value might be a list that would look like {"will","should","have","am","do","need"}, etc. The value that is associated with the key "I" is every word in Great Expectations that follows the word "I".

So Part A would be reading in Great Expectations into a HashMap, with the keys being the words in the file, and the values associated with each key are the words that follow a particular word throughout Great Expectations. More specifically, as you read in Great Expectations, the current word you are reading in is first added to the value set of the previous word you read in, which is the key. The current word then becomes the key and you read in the next word, which is the value of that word.

Once you have finished reading in Great Expectations, you will create your own Dickins document and print out your new document to a new text file. Start with the very first word you read in from Great Expectations. Print that word to the new file. Then, that word is a key. Take the value set associated with that key, convert it to an ArrayList (this should be one line of code), and randomly choose a word in this array list. This will be the second word printed to the text file, and now becomes the key. Take its corresponding value set, convert it to an ArrayList, and randomly choose among those words the next word to print. This word now becomes the key. Continue this for a count of maybe 500 or until the value set's size is 0.

Now read your new document. Anything remotely resembling Dickins? Have you created a great new novel?

A couple of notes: Great Expectations is a long book. I've included both Great Expectations and just Chapter 1. Do all your initial testing on Chapter 1, or you'll go nuts waiting to see of Great Expectations in its entirety has been read in successfully. You may even want to shorten Chapter 1 for the initial reading in tests.

Note 2: Leave punctuation in. So, in other words, "end" and "end." are two different strings that should be added separately to a value set. This makes the resulting newly created file marginally more readable.

Note 3: If we really wanted to do this right, we'd make the values be an ArrayList of objects, and the objects would have two fields, a string and a count. The count would keep track of how often a particular string followed the key. This would allow us to eliminate words that only occurred once as anomalies.

Problem 3: (25 pts) Game Learning

(Note: The game description is long, but the actual code is fairly straightforward to write)

In the game of sticks there is a heap of sticks on a board. On their turn, each player picks up 1 to 3 sticks. The one who has to pick the final stick will be the loser.

Sample game

The following is an example of the game of sticks.

- The game starts with 20 sticks on the board.
- Marvin takes 3 sticks, there are 17 sticks remaining.
- Hal takes 2 sticks, there are 15 sticks remaining.
- Marvin takes 1 stick, there are 14 sticks remaining.
- Hal takes 3 sticks, there are 11 sticks remaining.
- Marvin takes 2 sticks, there are 9 sticks remaining.
- Hal takes 2 sticks, there are 7 sticks remaining.

- Marvin takes 3 sticks, there are 4 sticks remaining.
- Hal takes 1 stick, there are 3 sticks remaining.
- Marvin takes 2 sticks, there is 1 stick remaining.
- Hal has to take the final stick and loses.

This assignment is split into three parts:

1. Implementing the game as a two-player game.
2. Adding an AI that can be played against.
3. Adding an option for training the AI against another AI.

Part one: Human vs. Human

First, create a game where two players can play against each other. The two examples below demonstrate how the game should behave.

Example 1

Welcome to the game of sticks!
How many sticks are there on the table initially (10-100)? 10

There are 10 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 7 sticks on the board.
Player 2: How many sticks do you take (1-3)? 3

There are 4 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There is 1 stick on the board.
Player 2: How many sticks do you take (1-3)? 1
Player 2, you lose.

Example 2

Welcome to the game of sticks!
How many sticks are there on the table initially (10-100)? 500
Please enter a number between 10 and 100
How many sticks are there on the table initially (10-100)? 3
Please enter a number between 10 and 100
How many sticks are there on the table initially (10-100)? 50

There are 50 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 47 sticks on the board.
Player 2: How many sticks do you take (1-3)? 55
Please enter a number between 1 and 3
Player 2: How many sticks do you take (1-3)? 3

There are 44 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3
...

There is 1 stick on the board.
Player 1: How many sticks do you take (1-3)? 1
Player 1, you lose.

Implement a game with the functionality described above.

Part two: Human vs. AI

Playing against your friends is nice, but you're studying computer science for a reason. We can learn to do some pretty cool stuff. So can the computers -- when taught. Let's create an AI for our game.

One way to create an AI for the game of sticks is to mathematically analyze the game and craft an AI based on the analysis. However, we now do the opposite. We create a learning AI that is able to learn a good strategy for the game of sticks by playing the game against us. After this, the mathematical analysis is easier to do because we already know how to play the game optimally.

Consider the functionality of the AI using the following description:

- An AI has a number of hats, one hat for each possible amount of sticks on the table. Initially, each hat contains three balls that are numbered from 1 to 3. If there are 10 sticks, there are 10 hats, each with 3 balls.
- At every step of the game that the AI plays, the AI takes a random ball out of the hat that matches the amount of sticks currently on the board. (So if the first player took 2 sticks, there are 8 sticks on the board. The AI takes a random ball (ball 1, 2, or 3), places it next to hat 8, and then reduces the number of sticks on the board by that random ball's amount.
- If the AI wins the game, it puts two balls of the type to each hat that has a ball next to it. Both balls have the same number. If the AI loses, it will throw away the balls that are next to the hats (note: A hat must always have at least one ball of each number, hence the last ball of a specific number cannot be thrown away and must be put back to the hat).
- As more and more games are played, there will be more balls that indicate a good number of sticks to take. This means that as balls are taken at random, it becomes more likely that the AI is able to play well.

Example

Here's an example where there are 10 sticks at the beginning. The hat contents for the AI are as follows:

hat	1	2	3	4	5	6	7	8	9	10
content	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3

The game may proceed as follows:

1. Player takes 3 sticks, there are 7 sticks remaining.
2. AI randomly picks up ball 2 from the hat 7. This means that the AI takes 2 sticks, and there are 5 sticks remaining.
3. Player takes 1 stick, there are 4 sticks remaining.
4. AI randomly picks up ball 3 from hat 4. This means that AI takes 3 sticks, and there is 1 stick remaining.

5. Player has to take the final stick and loses.

Now, the situation with the AI is as follows:

hat	1	2	3	4	5	6	7	8	9	10
content	1,2,3	1,2,3	1,2,3	1,2	1,2,3	1,2,3	1,3	1,2,3	1,2,3	1,2,3
beside				3			2			

As the AI wins the game, it will put the balls that are next to the hats back to the hats with extra balls. The situation is now as follows:

hat	1	2	3	4	5	6	7	8	9	10
content	1,2,3	1,2,3	1,2,3	1,2,3,3	1,2,3	1,2,3	1,2,2,3	1,2,3	1,2,3	1,2,3

Note: for the content of the hat, you should have some form of collection. The collection must allow doubles, and it must easily expand and contract in size. You will either be adding to the end of the collection, or removing from somewhere in the collection. Which collection most likely makes the most sense?

Now the AI will more likely take 3 sticks in the case of four sticks remaining on the board, and 2 sticks in case there are 7 sticks remaining on the board.

Your task is to modify the human vs. human version of the game so that the player can choose to play against an AI that works as described above. After each game, the AI will update the contents of its hats. The AI will play relatively randomly at first, but you will notice that it will start to learn a strategy as you play against it.

The following example displays how the program should behave after you have finished this step.

```
Welcome to the game of sticks!
How many sticks are there on the table initially (10-100)? 10
Options:
 Play against a friend (1)
 Play against the computer (2)
Which option do you take (1-2)? 2

There are 10 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 7 sticks on the board.
AI selects 2

There are 5 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 2 sticks on the board.
AI selects 2
AI loses.
Play again (1 = yes, 0 = no)? 1

There are 10 sticks on the board.
Player 1: How many sticks do you take (1-3)? 1

There are 9 sticks on the board.
```

```
AI selects 1

There are 8 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 5 sticks on the board.
AI selects 3

There are 2 sticks on the board.
Player 1: How many sticks do you take (1-3)? 2
You lose.
Play again (1 = yes, 0 = no)? 1

There are 10 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 7 sticks on the board.
AI selects 2

There are 5 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 2 sticks on the board.
AI selects 2
AI loses.
Play again (1 = yes, 0 = no)? 0
```

Part three: AI vs. AI

In the previous part we created an AI that is able to learn from playing against the player. As we play against it, we notice that it takes a considerable amount of time before the AI is able to perform against a human player. In this assignment, you need to modify the program so that the player can choose to play either against a naive AI or a pre-trained AI.

In order to pre-train an AI, you need to create a program that allows two AIs to battle against each others -- say a hundred thousand times (after the training is working, try out different numbers as well!) -- and after that the player will be set to play against the AI that is ready to battle the player.

The following example shows how the game would work with the trained AI option.

```
Welcome to the game of sticks!
How many sticks are there on the table initially (10-100)? 10
Options:
 Play against a friend (1)
 Play against the computer (2)
 Play against the trained computer (3)
Which option do you take (1-3)? 3
Training AI, please wait...

There are 10 sticks on the board.
Player 1: How many sticks do you take (1-3)? 3

There are 7 sticks on the board.
AI selects 2
```

```
There are 5 sticks on the board.
Player 1: How many sticks do you take (1-3)? 1

There are 4 sticks on the board.
AI selects 3

There is 1 stick on the board.
Player 1: How many sticks do you take (1-3)? 1
You lose.
Play again (1 = yes, 0 = no)? 1

There are 10 sticks on the board.
Player 1: How many sticks do you take (1-3)? 2

There are 8 sticks on the board.
AI selects 3

There are 5 sticks on the board.
Player 1: How many sticks do you take (1-3)? 2

There are 3 sticks on the board.
AI selects 2

There is 1 stick on the board.
Player 1: How many sticks do you take (1-3)? 1
You lose.
Play again (1 = yes, 0 = no)? 0
```

Congratulations! You've just taught the computer to learn.