Due: Thursday, Oct. 15, 2015 by 9pm. Late turn-in is Tuesday, Oct. 20, 2015 by 9pm.

Be sure to include the appropriate collaboration statement as comments at the top of your submitted program. Failure to do so will result in a 10 point deduction.

Move into your CS170 directory:
```
cd cs170
```
Make a directory for hw3:
```
mkdir hw3
```
And move into that directory:
```
cd hw3
```
You can now start up gedit:
```
gedit &
```

Name your program `HW3.java`. Write the following methods. Follow the specified naming conventions EXACTLY (capitalization, parameter order, etc).

Strategy: I HIGHLY recommend you write one method at a time, test it, and verify it works before moving on to another method. While it is not required, I also recommend you write some of these methods with while loops and some with for loops to get practice with both of them.

1. **Closest Bus Stop:**
   You are writing a method which calculates where the nearest bus stop is in a town. In this town, buses stop every 8th street, starting at 0th Street (so 0th street, 8th street, 16th street, and so on). Write a method named `closestBusStop` which takes a non-negative integer parameter indicating a street number. This function should return an integer indicating the street of the closest bus stop. Ties go to the lower street number. For example, the closest bus stop to 12th Street is 8th Street. Examples:
   ```
closestBusStop(1) → 0
closestBusStop(4) → 0
closestBusStop(5) → 8
   ```

2. **Number of Nickels:**
   You are writing a method to make change for a customer after a transaction in a shop. Write a method named `numNickels` which takes a given an non-negative amount (in pennies) as the integer input parameter. The method returns the number of nickels that would be given. Change is made by first determining the number of quarters, then dimes, then nickels, then pennies. This function returns the number of nickels in correctly-made change. Examples:
   ```
numNickels(40) → 1
numNickels(36) → 0
   ```

3. **Lottery Winner:**
   You are in charge of writing some software for the Georgia lottery commission to validate winning ticket numbers. Unbeknownst to most people, there's a hidden pattern in numbers which validates whether or not a number is a winning number. We can break a 6 digit number down as: X Y ZZZZ. If the first digit (X) occurs the number of times specified by the second
digit \(Y\) in the remainder of the number (marked with \(Z\)'s), then the number is a winning number. Otherwise it is not. Write a method named \texttt{ticket} which takes a 6 digit integer as input. The method should return a boolean value indicating whether or not the number is a winning number. The method should return \texttt{false} if the number is a not a 6 digit number or is a negative number. Examples:

\[
\texttt{ticket(923995)} \rightarrow \text{true} \text{ (because 9 occurs two times in the portion 3995)} \\
\texttt{ticket(923945)} \rightarrow \text{false} \text{ (because 9 occurs once, not twice, in the portion 3945)} \\
\texttt{ticket(4566)} \rightarrow \text{false} \text{ (not a 6 digit number)}
\]

4. **Triples**

You are playing a dice game (with a fair/unbiased die) that involves rolling 3 times and receiving a certain number of points. Write a method named \texttt{triples} which takes 3 input parameters, each representing a roll of a die. It should return the score for those three rolls based on the following rules:

1. If all three rolls/numbers are the same, the player receives 100 points.
2. If the player rolls two 1's, two 3's or two 6's (but not 3!), the user receives 45 points.
3. If all three rolls are odd or all three rolls are even (but don't meet the criteria of the two rules above), the user receives 20 points.
4. If two of the rolls sum to the other roll (and don't meet the criteria of the three rules above), then the user receives 10 points.
5. All other rolls return a score of 0.

Examples:

\[
\texttt{triples(4,4,4)} \rightarrow 100 \text{ (by rule 1)} \\
\texttt{triples(3,3,6)} \rightarrow 45 \text{ (note that rule 2 supersedes rule 4 here)} \\
\texttt{triples(2,2,4)} \rightarrow 20 \text{ (note that rule 3 supersedes rule 4 here)} \\
\texttt{triples(1,3,4)} \rightarrow 10 \text{ (by rule 4)}
\]

5. **Loaded Die**

In Lab4, you wrote a method to simulate a single roll of a fair die. That is, the probability of rolling any face of the die was equally likely (a 16.67% chance of any arbitrary face). In this method, you'll write a method to simulate the roll of a loaded, 6-sided die. The method, named \texttt{loaded}, takes no input parameters and returns an integer indicating what the result of the roll was. In other words, it will return a value between 1 and 6 inclusive, given the restrictions below.

1. The die should roll a 1 25% of the time
2. The die should roll a 2 25% of the time
3. The die should roll a 3 20% of the time
4. The die should roll a 4 15% of the time
5. The die should roll a 5 10% of the time
6. The die should roll a 6 5% of the time

Comments: Make sure your code is well-commented. You must comment your methods with a description of the purpose, description of the arguments required, and a description of the return value. You can and should also use comments throughout other areas of code. Remember that the purpose of comments is not to restate the code.

Submission:

1. Make sure to include your collaboration statement as a comment at the top of your file.
2. You may submit as many times as you wish; we only grade the last submission. We will not grade or consider early submissions.
3. Submit your .java file (not your .class or other files) to Blackboard for the HW3 assignment.