CS 1301 - Homework 07

Due before October 10, 2:02 *am* on Vista (WebCT). You can resubmit your work up until the deadline.

1. **Commission.java** –

   a. Write a method, *computCommission* that accepts the dollar amount of burgers sold and hotdogs sold. If the total amount sold is more than $500.00, then the commission is 25%. Otherwise, the commission is determined individually by the dollar amount of burgers sold and the dollar amount of hotdogs sold:

Burgers Sales	Commission
less than $100	5%
at least $100 but less than $200	7%
$200 or more	10%

Hotdog Sales	Commission
less than $50	5%
at least $50 but less than $150	10%
$150 or more	12%

   b. Solve the following problem using your method. Prompt the user to enter the total sales of burgers and hotdogs. Display the commission.

2. **Triangle2.java** – Do problem 5.19 from the text. Your solution to Homework 02, Problem 4 might be useful.

3. **CubeRoot.java** –

   a. Write a method, *guess* that accepts a floating point number and finds the smallest integer, raised to the 3rd power, which is less than the number. For instance, If the input is 86.4, the output is 4 because $4^3 = 64 < 86.4$.

   b. Write a method, *cubeRoot* that accepts a floating point number, *n* and an *initial guess*. The method will compute an approximation to the cube root of the number by repeatedly using the following formula (Newton's Method):

$$nextGuess = lastGuess - \frac{lastGuess^3 - n}{3 * lastGuess^2}$$

   The initial guess should be "close" for this method to work and you will use the result from *guess()* in part *a*. This value will be the starting value for *lastGuess*. If the difference between *nextGuess* and *lastGuess* is less than a very small number, such as 0.0001, you can claim that *nextGuess* is the approximated cube root of *n*. If not, *nextGuess* becomes the *lastGuess* and the process repeats.

   c. You can test *cubeRoot* like this:

   num = Read number from user

   Print: cubeRoot(num, guess(num))

   d. Write a program that reads a decimal number from the user and computes the approximated cube root. Display both the number and the approximated cube root, formatted with 4 decimals:

```
The cube root of 86.4000 is 4.4213
```

4. Problem 5.27 from the text.

NOTES:

a. **The problem incorrectly states that 17 and 71 are examples of palindromic primes**. The sample output shown is correct.

b. The methods written in HW 06, problems 3 and 4 should be used.

c. You should write an additional method, *isPalindrone* which determines if a number is a palindrone. Hint: this will use the methods from HW 06, problem 3.

d. The algorithm (without considering printing 10 per line) is:

```
i=2
While( count < 100 )
      If( isPalindron(i) and isPrime(i) )
            Increment count
            Print i
      Increment i
```