Modeling the Metabolism of Ethanol to More Accurately Predict Blood Alcohol Content Part II

New Mexico Adventures in
Supercomputing Challenge
Final Report
April 6, 2004

Team 002
Alamogordo High School

Levi Blackstone
Matthew Woller

Ms. Dittrich

Kevin Blackstone

Table of Contents:

Executive Summary:

The purpose of this project was to create a mathematical model to simulate the removal of alcohol from the human body in order to obtain a blood alcohol content (BAC) prediction. An accurate prediction of BAC would be useful in a variety of fields. Lawyers, police officers, scientists, and the average person could use such predictions in many different circumstances.

Compartmental modeling is a method of mathematical simulation where various parts of a system are represented as a "compartment" and equations are used to model transfer between compartments. We hypothesized that a compartmental model representing the stomach, small intestine, and lean body mass could provide a more accurate estimate of BAC than current models.

We are continuing this project from last year in an effort to improve the accuracy of the simulation. We have updated the stomach compartment by including an equation that accounts for first-pass metabolism (elimination of alcohol in the stomach) and have modified our rate equations between compartments. Additionally, we are allowing for a variable liquid volume in the stomach and intestine to obtain a more realistic simulation.

We wrote a computer program in C++ to implement our compartmental model and compare the accuracy of our predictions to those of currently existing models. We chose to rewrite the program in Java this year to allow the implementation of a sophisticated Graphical User Interface (GUI) that can be used on any operating system. We restructured the program in this process in a more efficient, object-oriented fashion.

We obtained experimental data from the Texas Transportation Institute as a benchmark for the accuracy of our predictions. We concluded that our compartmental model provides a more accurate prediction for both male and female subjects.

Introduction:

Predicting blood alcohol content (BAC) and the elimination rate of alcohol could be useful in a variety of circumstances. Individuals wishing to estimate their blood alcohol content after a certain time could benefit from such a prediction as well as professionals such as bartenders, police officers, and lawyers. Perhaps a more important application would be to scientists investigating the effects of alcohol on human health by providing a guide for effective dosing methods.

Several general models for BAC predictions are currently used by the Traffic Department that use an average blood alcohol curve to estimate ethanol elimination. These models do not provide an accurate prediction in a majority of circumstances because of variations in a number of factors for each individual. Most general models merely account for the body weight and number of drinks consumed by the individual, ignoring individual height, age, gender, drinking history, amount of food in the stomach and a host of other variables affecting the accuracy of the prediction. These models also use a general elimination curve to calculate alcohol elimination instead of modeling the process of absorption and elimination. Using the overall body mass in predictions creates especially inaccurate results because alcohol is only distributed in the lean body mass. People with a higher percentage of body fat reach higher blood alcohol contents after consuming the same dose of alcohol than people with a lower percentage of fat.

Two recently developed computer models are used to predict BAC (cBAC; Addiction Research Foundation, London, Ontario, Canada, 1991; and BACest; National Highway Traffic Safety Administration, Washington, DC, 1994). The cBAC model uses height, weight, gender, and (for men only) age to estimate total body water (TBW), and the BACest program uses only body weight and gender as variables, using a separate percentage of body weight for men and

women to determine TBW. In a recent study, Davies and Bowen (2000) tested these models to determine the accuracy of the predicted peak BAC's compared to actual experimental data. They found that each model seriously underestimated the peak BAC's for their group of test subjects.

We believe that the process of ethanol metabolism can be modeled to provide more accurate estimates of blood alcohol content over time, and may overcome some of the problems posed by general models.

Materials and Methods:

We conducted a review of current literature on ethanol pharmokinetics, and learned from Kapur (1991) that most existing BAC prediction models are based on Widmark's (1932) mathematical equation (Appendix A). We also discovered that Watson *et al.* (1980) had developed regression equations to calculate total body water (TBW) to make an accurate estimation of the actual distribution volume of alcohol for each individual; this was the most notable update to Widmark's work (Appendix A). Pieters *et al.* (1990) developed a three-compartment model (stomach, small intestine, lean body mass) to simulate ethanol metabolism (Appendix A). Their model assumed that first-pass metabolism was insignificant and that ethanol is released from the stomach at approximately a first-order rate (rate of transfer to the small intestine is dependent on the concentration of alcohol in the stomach). This model also assumed that ethanol absorption from the small intestine followed a first-order rate and that ethanol metabolism from the lean body mass follows Michaelis-Menton kinetics (Appendix A).

We based our new model on the information from our research. Our model uses three compartments (stomach, intestine, lean body mass) to implement our equations. In the stomach compartment, we account for changes in liquid volume based on gastric secretions and the volume of alcohol being consumed. Ethanol leaves the stomach compartment either through first-pass-metabolism, simulated by a Michaelis-Menton kinetics equation, or through the pyloric sphincter to the small intestine at a first-order rate. We assume that the volume of liquid in the stomach leaves at a zero-order rate (constant rate) because of feedback inhibition in the small intestine controlling the rate of release. In the small intestine, we assume that the ethanol is absorbed into the lean body mass at a first-order-rate. Liquid is also assumed to leave the small intestine at a first-order-rate as it is absorbed by the intestines. Finally, we assume that ethanol is

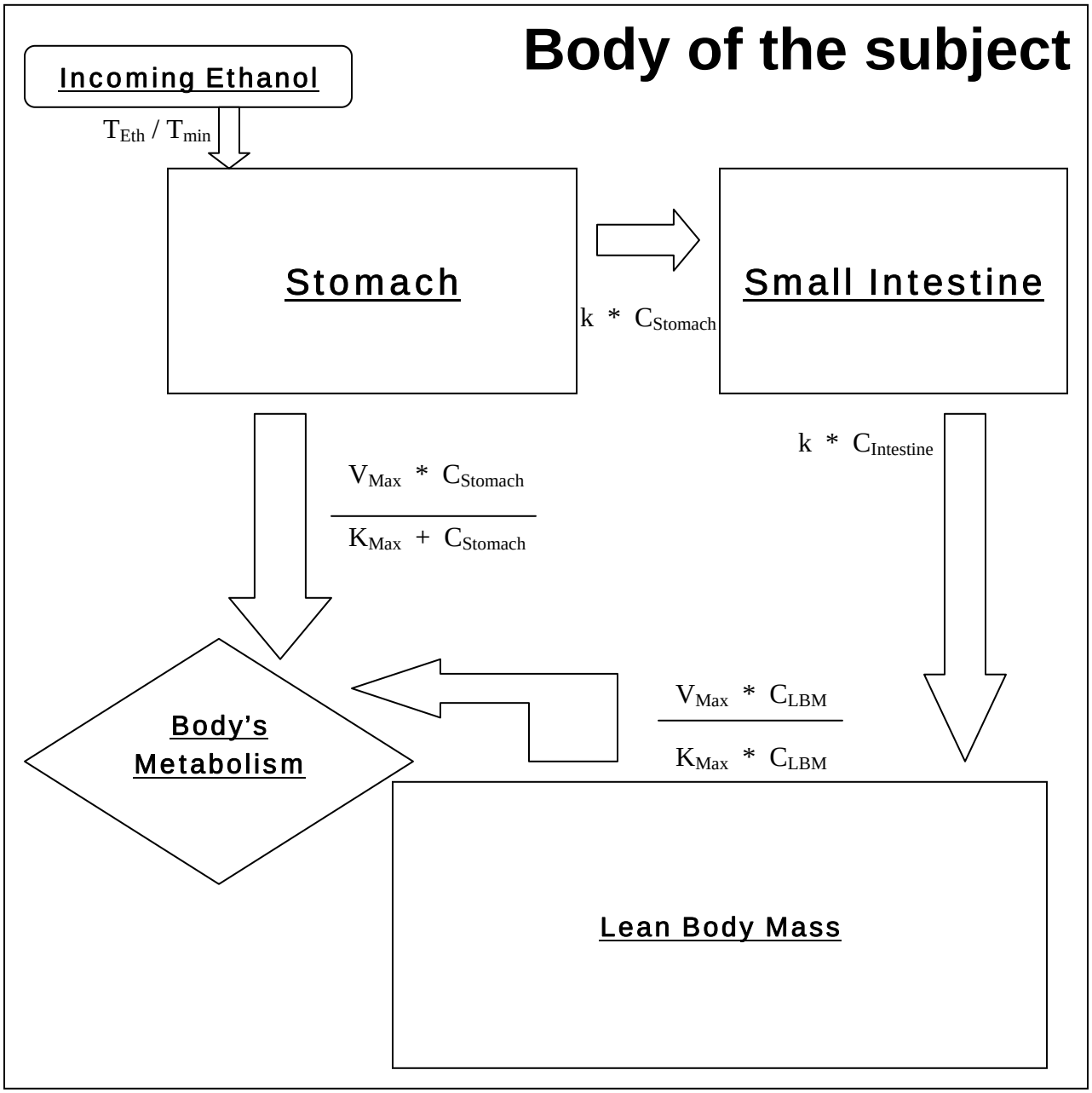
eliminated from the lean body mass following Michaelis-Menton kinetics, being dissolved in a liquid volume equal to the TBW. We use Watson *et al.* TBW equations to calculate TBW in our model. We considered modeling the transport of ethanol through the bloodstream (allowing ethanol to return to compartments it had left previously), but ultimately decided not to include this factor in our model because of the difficulty of obtaining specific blood flow rate values. We felt that the inclusion of this factor without reliable data would adversely affect the accuracy of our predictions, rather than improving them.

Our model last year did not account for several factors important in the process of ethanol metabolism. Previously, we ignored first-pass metabolism in the stomach, and assumed that it was merely a zero-order transport compartment (serving only to delay the absorption of ethanol into the body). We also ignored the presence of food in the stomach and the drinking history of the subject, which both affect the rate of ethanol absorption into the body. Additionally, we did not calculate separate liquid volumes for each compartment, instead dividing the amount of ethanol in each compartment by the TBW to calculate concentration (consequently affecting our transfer rates).

We developed a computer program to test our model in an efficient fashion and allow a user to input values for each subject to be tested. To estimate errors associated with the prediction model, we used a root-mean-square calculation, a statistical method of finding deviation from a mean value (see Results).

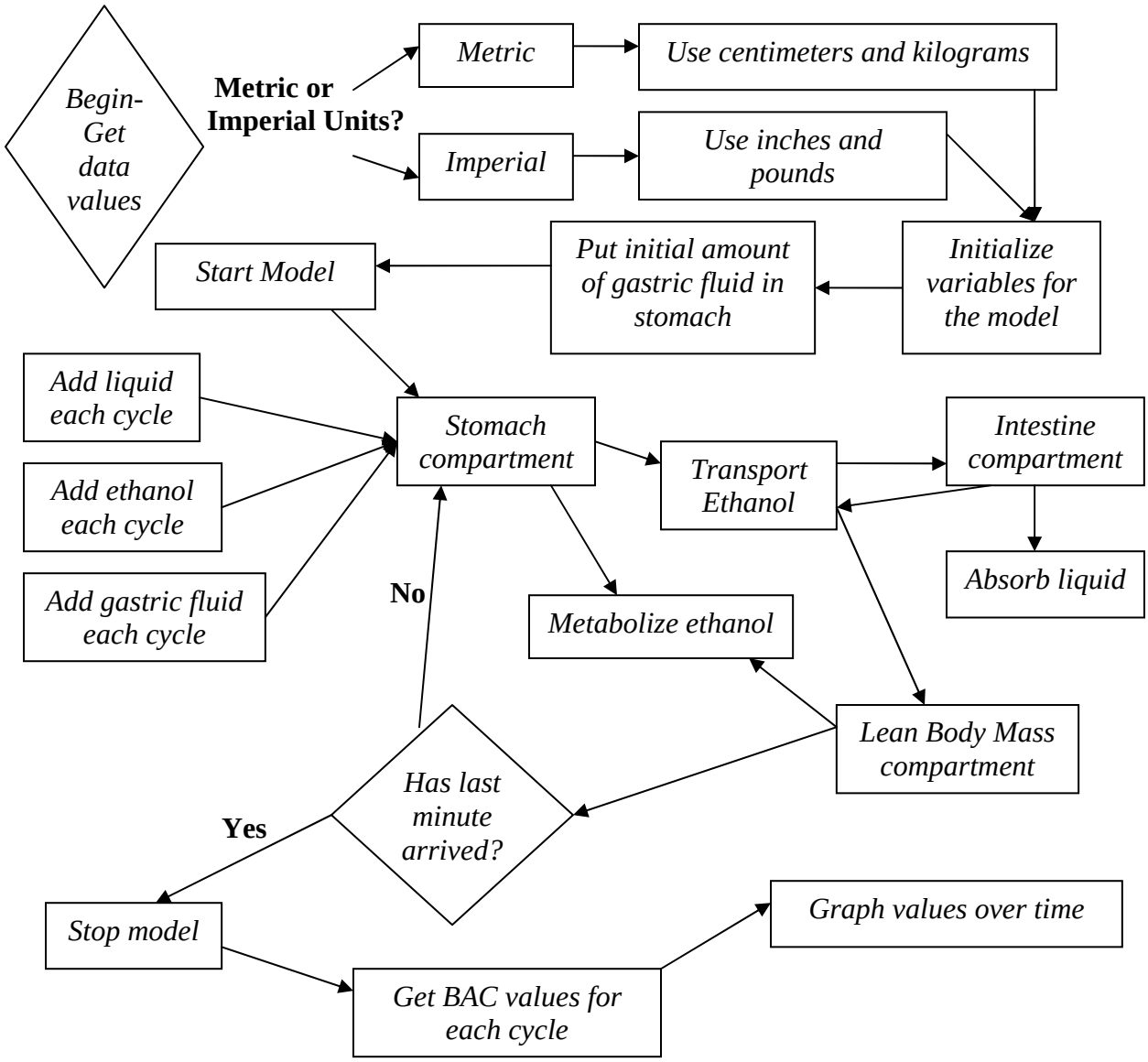
Model:

The following is a visual representation of the compartmental model used in our project. Constant values are omitted for simplification.



Program:

This chart shows the processes and logic used by our program to implement the model.

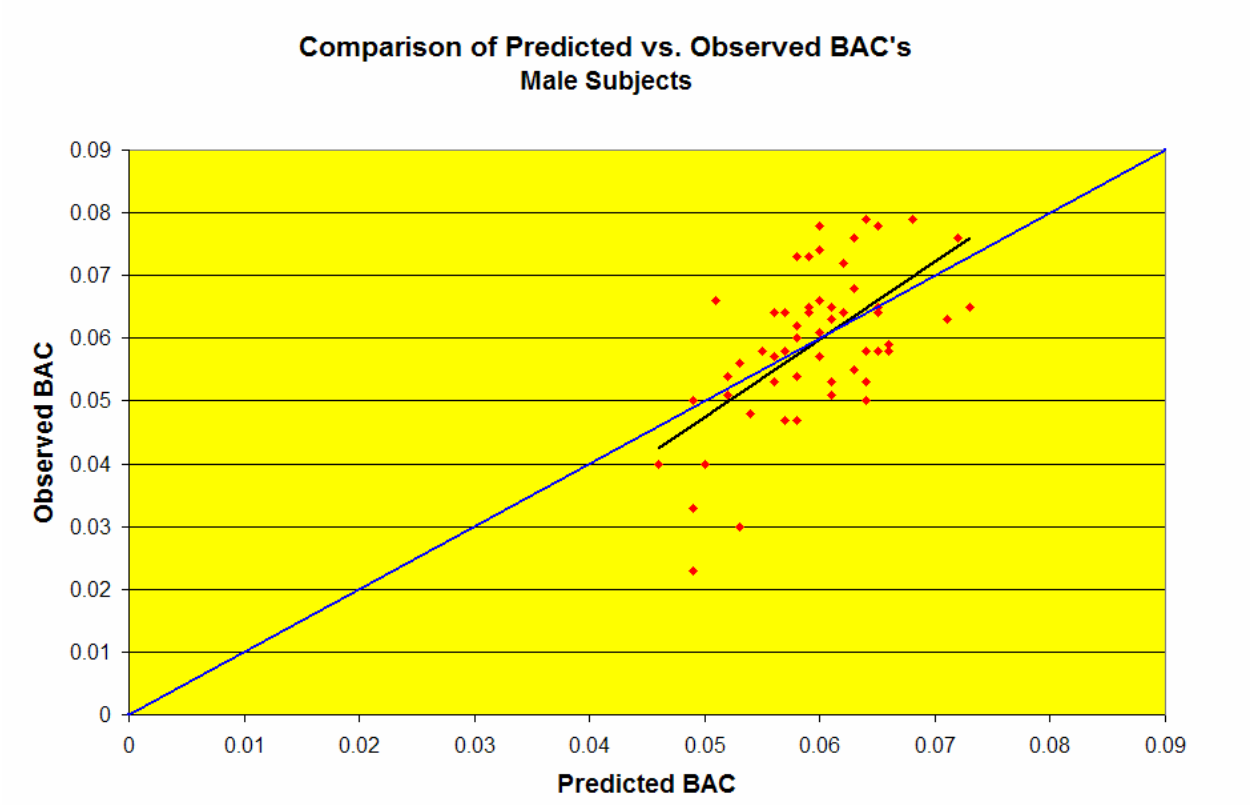


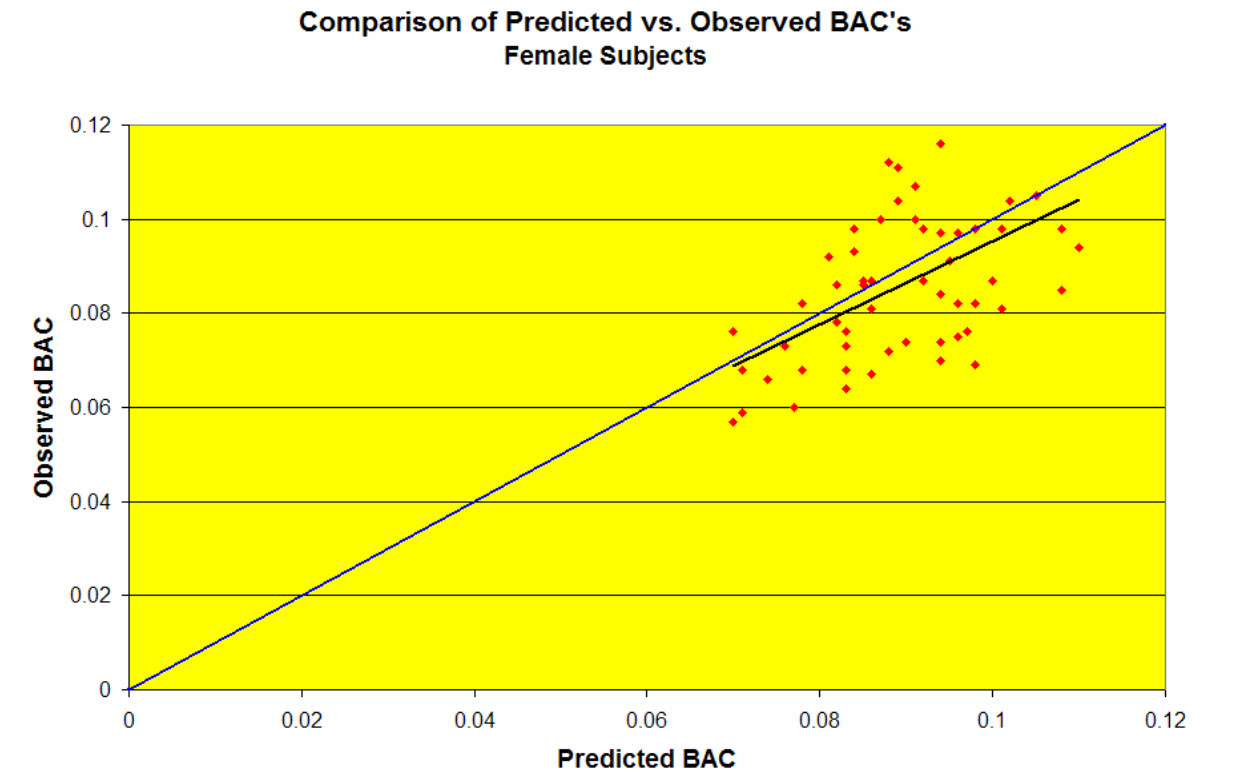
Results:

Our model this year has produced more accurate results than both last year's model and other models like BACest and cBAC, as demonstrated by a root-mean-square analysis (lower numbers are better because they represent values closer to the actual value).

Root-Mean-Square Analysis of Prediction Values		
Models	Blood Alcohol Content	
	Males	Females
Our Current Model	0.009	0.014
Our Model in 2002-2003	0.011	0.017
BACest	0.015	0.017
cBAC	0.013	0.016

The following graphs plot our model's predictions compared to the actual value for each subject. For an exact prediction, the dot will lie on the blue line. If the prediction overestimates the BAC, the dot will lie below the line, and above the line if the program underestimates the BAC. The black line is a linear regression of the data.


Comparison of Predicted vs. Observed BAC's
Male Subjects
Observed BAC
Predicted BAC
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09


Comparison of Predicted vs. Observed BAC's
Female Subjects
Observed BAC
Predicted BAC
0
0.02
0.04
0.06
0.08
0.1
0.12

Discussion:

The results support the hypothesis that a compartmental model can be used to more accurately predict BAC. The root-mean-square analysis of the data showed that the accuracy of the predictions made with the new model was better in both male and female subjects. A scatter plot of the data shows that our model exhibited no apparent trend of over or underestimation in male subject. The predictions for females showed a slight trend of overestimation. These results are more accurate than the predictions from the cBAC and BACest programs tested by Davies and Bowen (2000).

Recommendations:

To further improve the accuracy of the predictions of the new model, we need to acquire more data like that from the Texas Transportation Institution. By comparing our results to an even wider variety of individuals, we could easily determine specific groups of people (i.e. certain weights, ages, etc.) that our program over or underestimates for, and alter our equations appropriately. Additionally, we need to find additional literature to confirm the rate equations we use in our model and allow us to account for blood flow between compartments.

A statistical analysis could be implemented to determine the probability of our prediction lying within the range of measured BAC values for subjects with similar inputs (age, height, weight, etc). Finally, to determine its significance to the accuracy of the predictions, the time step used in the program (one minute increments) may be shortened, allowing the efficiency versus accuracy to be studied statistically.

Acknowledgements:

Thank you to Kevin Blackstone for his assistance in obtaining research and for suggestions on all biological aspects of the project. Thanks to Ms. Sharon Dittrich for her assistance with our presentation and suggestions for improving our project. Many thanks to Becky Davies from the Texas Transportation Institute for providing experimental data used to test the accuracy of our predictions. Thanks to the AiS preliminary judges for many helpful suggestions on our presentation.

References:

E. Baraona. "Site and Quantitative Importance of Alcohol First-Pass Metabolism." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 405-406.

B.T. Davies, M.A., and C.K. Bowen, M.S. "Peak Blood Alcohol Prediction: An Empirical Test of Two Computer Models". *J Alc Stud*. 2000; 61:1

B.T. Davies, M.A., and C.K. Bowen, M.S. "Total Body Water and Peak Alcohol Concentration: A Comparative Study of Young, Middle-Age, and Older Females." *Alcoholism: Clinical and Experimental Research* Vol. 23, No. 6, 1999. pp 969-975.

R.T. Gentry. "Determinants and Analysis of Blood Alcohol Concentrations After Social Drinking." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. p 399.

R.T. Gentry. "Effect of Food on the Pharmacokinetics of Alcohol Absorption." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 403-404.

P.S. Haber. "Metabolism of Alcohol by the Human Stomach." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 407-408.

A.W. Jones. "Aspects of In-Vivo Pharmacokinetics of Ethanol." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 400-402.

H. Kaliant. "Effects of Food and Body Composition and Blood Alcohol Curves." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 413-414.

B.M. Kapur. "Alcohol: Pharmacology, methods of analysis and its presence in the casualty room." In *Drinking and Casualties: Accidents, poisonings and violence in an international perspective*, Eds: N Giesbrecht, R Gonzalez, M. Grant et al., Tavistock/Routledge, London/New York, 1991; pp 172-187.

B.M. Kapur. "CBAC: Computerized Blood Alcohol Concentration – A Computer Model as a Clinical and an Educational Tool." <http://www.clinitox.com/cbac/cbac1.htm>.

T.K. Li, J.D. Beard, W.E. Orr, P.Y. Kwo, V.A. Ramchandani, H.R. Thomasson. "Variation in Ethanol Pharmacokinetics and Perceived Gender and Ethnic Differences in Alcohol Elimination." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 415-416.

C.S. Lieber. "Ethnic and Gender Differences in Ethanol Metabolism." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 417-418.

M.D. Levitt and D.G. Levitt. "Use of a Two-Compartment Model to Assess the Pharmacokinetics of Human Ethanol Metabolism." *Alcoholism: Clinical and Experimental Research* Vol. 22, No. 8, 1998. pp 1680-1688.

M.D. Levitt and D.G. Levitt. "Use of a Two-Compartment Model to Predict Ethanol Metabolism." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 409-410.

E. Mezey. "Influence of Sex Hormones on Alcohol Metabolism." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. p 421.

J.E. Pieters, M. Wedel and G.J. Schaafsma. "Parameter estimation in a three-compartment model for blood alcohol curves". *Alcohol and Alcoholism* 25, 17-24 (1990). pp 17-24.

R. Roine. "Interaction of Prandial State and Beverage Concentration of Alcohol Absorption." *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 411-412.

H. Thomasson. "Alcohol Elimination: Faster in Women?" *Alcoholism: Clinical and Experimental Research* Vol. 24, No. 4, 2000. pp 419-420.

P.E. Watson, I.D. Watson and R.D. Batt. "Prediction of blood alcohol concentrations in human subjects. Updating the Widmark equation". *J Alc Stud.* 1981; 42:547-556.

M. Wedel, J.E. Pieters, N.A. Pikaar, Th. Ockhuizen. "Application of a Three-Compartment Model to a Study of the Effects of Sex, Alcohol Dose and Concentration, Exercise and Food Consumption on the Pharmacokinetics of Ethanol in Healthy Volunteers." *Alcohol & Alcoholism* Vol. 26, No. 3. pp 329-336, 1991.

D. Whitmire, L. Cornelius, P. Whitmire. "Monte Carlo Simulation of an Ethanol Pharmacokinetic Model." *Alcoholism: Clinical and Experimental Research* 26, No. 10, 2002: pp 1484-1493.

E.M.P. Widmark. "Die theoritischen Grundlagen und die praktische Verwendbarkeit der gerichtlichmedizinischen Alkoholbestimmung". Berlin; Urban & Schwarzenberg; 1932. (Widmark EM. in *Principles and applications of medicolegal alcohol determination*; Translated from 1932 german edition, Biomedical Publications, Davis, CA. 1981.

Appendix A

Widmark Formula

$$\left[\frac{(n)(d)(0.0514)(1.055)}{(w*R)}*100\right]-(B*t)$$

n = number of drinks d = oz. ethanol/drink w = subject weight B = hourly decrease
(0.0514) = lbs. ethanol/oz. drink (1.005) = g ethanol/mL
R = Widmark "R" value – percentage of body mass containing alcohol (total body mass – mass of fat and bone)

Watson TBW (males)

$$\left[(-0.09516*a)+(0.1074*h)+(0.3362*w)\right]+2.447$$

a = age in years h = height in cm w = weight in kg

Watson TBW (females)

$$\left[(0.1069*h)+(0.2466*w)\right]-2.097$$

h = height in cm w = weight in kg

Formula Derived From Michaelis-Menton Equation

$$-\frac{(V_m C_3)}{(K_m + C_3)}$$

V_m = maximum elimination rate when the alcohol-oxidizing enzyme is saturated
K_m = concentration at which the elimination rate is ½ V_m
C_3 = concentration of ethanol in compartment three (lean body mass)

Pieters Rate Equations

$$\Delta C_1 = -k_1(1+aC_1^2)^{-1}C_1$$
$$\Delta C_2 = k_1(1+aC_1^2)^{-1} - k_2 C_2$$
$$\Delta C_3 = k_2 C_2 - \frac{V_m C_3}{(K_m + C_3)}$$

C_x = concentration for compartments 1, 2, 3
k_1 = rate constant for compartment one (stomach)
k_2 = rate constant for compartment two (small intestine)
a = feedback control (faster for neg. values, slower for pos. values)
V_m, K_m = *see Formula Derived From Michaelis-Menton Equation*

Root-Mean-Square Formula

$$\sqrt{\frac{\sum_{\text{Observations}} (\text{Observed BAC} - \text{predicted BAC})^2}{\text{Number of observations (subjects)}}}$$

Appendix B (code)

NMSubject.java

```
package metab;
import metab.*;
import java.io.*;


/**
 * The <code>NMSubject</code> class handles the bulk of the computations for the
 * compartmental model.  An instance of the class is representative of one human
 * subject.  The subject contains exactly one <code>Stomach</code>, <code>Intestine</code>,
 * and <code>LeanBodyMass</code> compartment.  See the constructor for the information
 * required for using this class.  Values for the bitFlag argurment need to be
 * or'd ( | ) together, with one option set for each category. The categories are
 * food present in the stomach, drinking history, and units of measurement.  When
 * _METRIC is selected as the units of measurement, cm and kg should be used for
 * the height and weight arguments, and when _IMPERIAL is selected, in and lbs
 * should be used.  When returning the bitFlag, the user should check to see if
 * the _ERROR bit is checked.  If so, then the user constructed the object incorrectly
 * and should re-initialize it appropriately.  This class keeps track of BAC during
 * each minute time interval and final BAC should be determined from the array
 * determined by the getBAC method.
 * @author Levi Blackstone, Matthew Woller
 * @version 1.29
 * @see metab.Stomach
 * @see metab.Intestine
 * @see metab.LeanBodyMass
 * @see metab.Subject
 * @see metab.Drink
 * @see metab.DrinkTime
 * @see metab.TDSubject
 */
public class NMSubject implements Serializable {

	/**
	 * Standard value representing a light meal was eaten recently.
	 */
	public static final int _LIGHT_MEAL = 1;
	/**
	 * Standard value representing an average meal was eaten recently.
	 */
	public static final int _AVERAGE_MEAL = 1 << 1;
	/**
	 * Standard value representing a large meal was eaten recently.
	 */
	public static final int _LARGE_MEAL = 1 << 2;
	/**
	 * Standard value representing no meal was eaten recently.
	 */
	public static final int _NO_MEAL = 1 << 3;
	/**
	 * Standard value representing that the subject is a light drinker.
	 */
	public static final int _LIGHT_DRINKER = 1 << 4;
```

```
/**
 * Standard value representing that the subject is a moderate drinker.
 */
public static final int _MODERATE_DRINKER = 1 << 5;
/**
 * Standard value representing that the subject is a heavy drinker.
 */
public static final int _HEAVY_DRINKER = 1 << 6;
/**
 * Standard value representing that input units are in metric units.
 */
public static final int _METRIC = 1 << 7;
/**
 * Standard value representing that input units are in Imperial units.
 */
public static final int _IMPERIAL = 1 << 8;
/**
 * Standard value representing that an error occured while retrieving info.
 */
public static final int _ERROR = 1 << 9;
/**
 * Amount of liquid exiting the stomach through pyloric sphincter (L/min).
 */
private static final double squirtVolume = 0.0111;
/**
 * Amount of liquid exiting the intestine through the villi (L/min).
 */
private static final double absorbVolume = 0.01;
/**
 * Amount of gastric fluid being produced in the stomach (L/min).
 */
private static final double newFluid = .002;

/**
 * Bitflag representing all set switches.
 */
private int bitFlag;
/**
 * Age in years.
 */
private int age;
/**
 * Represents number of minutes after the specified amount at which point
 * the BAC would become zero.
 */
private int timeForZeroEthanol;
/**
 * Total number of minutes in drinking interval.
 */
private int totalMinutes;
/**
 * Array of the value for the BAC at every time interval.
 */
private double BACvalues[];
/**
 * Height (cm).
 */
```

```
private double height;
/**
 * Weight (kg).
 */
private double weight;
/**
 * Represents the highest BAC achieved during the time interval.
 */
private double peakBAC;
/**
 * Represents the total amount of ethanol to be introduced (g).
 */
private double totalEthanol;
/**
 * Represents the total amount of liquid to be introduced (L).
 */
private double totalVolume;
/**
 * Represents the total water content of the body (L).
 */
private double TBW;
/**
 * Represents the amount of ethanol to be introduced each cycle (g/min).
 */
private double passingEthanol;
/**
 * Represents the amount of liquid to be introduced each cycle (L/min).
 */
private double passingVolume;
/**
 * Represents the gender of the subject ('M' or 'F').
 */
private char gender;
/**
 * Represents the name and/or description of the subject.
 */
private String subjectName;
/**
 * Array of all the drinks consumed by the subject.
 */
private Drink drinks[];
/**
 * Represents the time that the subject started consuming their drinks.
 */
private DrinkTime start;
/**
 * Represents the time that the subject stopped consuming their drinks.
 */
private DrinkTime end;
/**
 * Represents the subject's stomach.
 */
private Stomach stomach;
/**
 * Represents the subject's small intestine.
 */
private Intestine intestine;
```

```
	/**
	 * Represents the subject's non-fatty body mass.
	 */
	private LeanBodyMass body;


	/**
	 * Constructor for the New Model subject class.
	 * @param age <code>int</code> representing the age of the subject in years.
	 * @param height <code>double</code> representing the height of the subject.
	 * @param gender <code>char</code> representing the gender of the subject.
	 * @param weight <code>double</code> representing the weight of the subject.
	 * @param drinks <code>Drink</code> array representing all the drinks drank
	 *                         by the subject.
	 * @param start <code>DrinkTime</code> representing the time that the
	 *                         subject started drinking.
	 * @param end <code>DrinkTime</code> representing the time that the subject
	 *                         stopped drinking.
	 * @param subjectName <code>String</code> representing the subjects name and
	 *                         / or description.
	 * @param bitFlag <code>int</code> representing all set switches.
	 */
	NMSubject(int age, double height, char gender, double weight, Drink [] drinks,
	               DrinkTime start, DrinkTime end, String subjectName, int bitFlag){

	      this.age = age;
	      this.gender = gender;
	      this.drinks = drinks;
	      this.start = start;
	      this.end = end;
	      this.subjectName = subjectName;
	      this.bitFlag = bitFlag;
	      this.peakBAC = 0.00;

	      if ((bitFlag & _METRIC) != 0) {
	            this.weight = weight;
	            this.height = height;
	      }
	      else if ((bitFlag & _IMPERIAL) != 0) {
	            this.weight = weight / 2.20462262; // lb to kg
	            this.height = height * 2.54; // in to cm
	      }

	      if (this.gender == 'M'){
	      this.TBW = (-0.09516 * (this.age)) + (0.1074 * (this.height)) + (0.3362 *
(this.weight)) + 2.447;
	    }
	    else if (this.gender == 'F'){
	      this.TBW = (0.1069 * (this.height)) + (0.2466 * (this.weight)) - 2.097;
	    }
//System.out.println(this.TBW);

	      for (int i=0; i < drinks.length; i++){
	            this.totalVolume += drinks[i].getVolume() * 0.0295735297; // Liters
	            this.totalEthanol += (drinks[i].getVolume() *
(drinks[i].getConcentration()/100))
	                                          * 23.36; // Grams
```

```
            }
//System.out.println(this.totalVolume);
//System.out.println(this.totalEthanol);

            // if drinking period crosses 12 o' clock
            if ((end.getAMorPM() == DrinkTime._AM && start.getAMorPM() == DrinkTime._PM) ||
                  (end.getAMorPM() == DrinkTime._PM && start.getAMorPM() == DrinkTime._AM))
{
                  this.totalMinutes = (((12 - start.getHours()) * 60) + (0 -
start.getMinutes()))
                                                + (end.getHours() * 60) + end.getMinutes();
            } else {
                  if (start.getHours() == 12)
                this.totalMinutes = (end.getHours() * 60) + (end.getMinutes() -
start.getMinutes());
            else {
                this.totalMinutes = (end.getHours() - start.getHours()) * 60
                                                + (end.getMinutes() - start.getMinutes());
                  }
            }

            this.BACvalues = new double[this.totalMinutes];

            this.passingEthanol = this.totalEthanol / this.totalMinutes;
            this.passingVolume = this.totalVolume / this.totalMinutes;
//System.out.println(this.passingEthanol);
//System.out.println(this.passingVolume);

      }

      /**
       * Retrieves the age of the subject.
       * @return <code>int</code> representing the subject's age (yrs).
       */
      public int getAge() {

            return this.age;

      }

      /**
       * Retrieves the list of BACvalues recorded while the model was running.
       * One value will have been recorded for each minute the subject was drinking.
       * @return Array of <code>double</code>s containing the chronological list of
       * BAC values.
       */
      public double [] getBAC() {

            double BAC[] = new double[this.BACvalues.length];
            System.arraycopy(this.BACvalues, 0, BAC, 0, this.BACvalues.length);
            return BAC;

      }

      /**
       * Retrieves the options set in the subject's bit flag.
       * @return <code>int</code> representing the bit flag.
```

```
 */
public int getBitF() {

	return this.bitFlag;

}

/**
 * Retrieves the list of drinks consumed by the subject.
 * @return Array of <code>Drink</code>s containing the drink objects.
 */
public Drink [] getDrinks() {

	Drink drinks[] = new Drink[this.drinks.length];
	System.arraycopy(this.drinks, 0, drinks, 0, this.drinks.length);
	return drinks;

}

/**
 * Retrieves the time when the subject stopped drinking.
 * @return <code>DrinkTime</code> representing the end time.
 */
public DrinkTime getEndTime() {

	return this.end.getCopy();

}

/**
 * Retrieves the height of the subject depending on the system of
 * measurement selected.
 * @return <code>double</code> representing the height (cm or in).
 */
public double getHeight() {

	if ((bitFlag & _METRIC) != 0) {
		return this.height;
	} else if ((bitFlag & _IMPERIAL) != 0) {
		return this.height / 2.54; // cm to in
	} else {
		return 0;
	}

}
/**
 * Retrieves the gender of the subject.
 * @return <code>char</code> representing gender ('M' or 'F').
 */
public char getGender() {

	return this.gender;

}

/**
 * Retrieves the total number of minutes for the drinking time.
```

```
     * @return <code>int</code> representing the total number of minutes.
     */
    public int getMinutes() {

          return totalMinutes;

    }

    /**
     * Retrieves the name and description of the subject.
     * @return <code>String</code> containing the subject name.
     */
    public String getName() {

          return this.subjectName;

    }

    /**
     * Retrieves the peak BAC achieved during the drinking period.
     * @return <code>double</code> representing the peak BAC.
     */
    public double getPeakBAC() {

          return this.peakBAC;

    }

    /**
     * Retrieves the time when the subject stopped drinking.
     * @return <code>DrinkTime</code> representing the end time.
     */
    public DrinkTime getStartTime() {

          return this.start.getCopy();

    }

    /**
     * Retrieves the Total Body Water (TBW) for the subject.
     * @return <code>double</code> representing the TBW.
     */
  public double getTBW() {

       return this.TBW;

    }

    /**
     * Retrieves the weight of the subject depending on the system of
     * measurement selected.
     * @return <code>double</code> representing the weight (kg or lb).
     */
    public double getWeight() {

          if ((bitFlag & _METRIC) != 0) {
                return this.weight;
```

```
            } else if ((bitFlag & _IMPERIAL) != 0) {
                  return this.weight * 2.20462262; // cm to in
            } else {
                  return 0;
            }

      }

      /**
       * Checks to see if a new peak BAC has been reached.
       */
      private void peakBAC() {

            if (this.peakBAC < (body.getContents() / this.TBW) / 10) {
                  this.peakBAC = (body.getContents() / this.TBW) / 10;
            }

      }

      /**
       * Runs the compartmental model.  This method should only be run if all
       * values have already been declared and initialized, otherwise, unexpected
       * results may occur.
       * @return <code>boolean</code> value that represents whether the method
       *                executed successfully.  <code>true</code> if successful,
       *                otherwise, unsuccessful.
       */
    public boolean runModel() {

            /*
             * Initializes the stomach compartment.  Initial volume is equal to the
             * gastric fluid already present (depending on meal).
             */
            stomach = new Stomach(0.00,
                  ((bitFlag & NMSubject._NO_MEAL) != 0) ? .150 :
                  ((bitFlag & NMSubject._LIGHT_MEAL) != 0) ? .600 :
                  ((bitFlag & NMSubject._AVERAGE_MEAL) != 0) ? 1.000 :
                  ((bitFlag & NMSubject._LARGE_MEAL) != 0) ? 1.400 : 0.00);
            intestine = new Intestine(0.000, 0.000);
            int drinkingHistory = ((bitFlag & NMSubject._LIGHT_DRINKER) != 0) ?
NMSubject._LIGHT_DRINKER :
                                    ((bitFlag & NMSubject._MODERATE_DRINKER) != 0) ?
NMSubject._MODERATE_DRINKER :
                                    ((bitFlag & NMSubject._HEAVY_DRINKER) != 0) ?
NMSubject._HEAVY_DRINKER : NMSubject._MODERATE_DRINKER;
            body = new LeanBodyMass(0.000, this.TBW, drinkingHistory);

            int min = 0;
            while (min < this.totalMinutes) {

                  double firstPass = stomach.computeMMMetab() * .04607 *
stomach.getLiquid(); // grams of ethanol metabolized.
                  double toIntestine = stomach.computeEthanolRelease(); // grams of ethanol
leaving through pyloric sphincter.
                  double toBody = intestine.computeEthanolAbsorption(); // grams of ethanol
leaving through villi
                  double fromBody = body.computeMMMetab() * .04607 * this.TBW; // grams of
```

```
ethanol metabolized.
//System.out.println(this.passingEthanol);
//System.out.println(this.passingVolume);
//System.out.println(firstPass);
//System.out.println(squirtVolume);
//System.out.println(toIntestine);
//System.out.println(stomach.getLiquid());
//System.out.println(intestine.getLiquid());
//System.out.println(stomach.getContents());
//System.out.println(intestine.getContents());
//System.out.println(intestine.getContents());
//System.out.println(body.getContents());

                    // adjust stomach transfer values.
                    stomach.addContents(this.passingEthanol);
                    stomach.addLiquid(this.passingVolume);
                    stomach.addContents(-firstPass);
                    stomach.addLiquid(-this.squirtVolume);
                    stomach.addContents(-toIntestine);

                    // adjust intestine transfer values.
                    intestine.addContents(toIntestine);
                    intestine.addLiquid(this.squirtVolume);
                    intestine.addContents(-toBody);
                    intestine.addLiquid(-this.absorbVolume);

                    // adjust body transfer values.
                    body.addContents(toBody);
                    body.addContents(-fromBody);

                    BACvalues[min] = (body.getContents() / this.TBW) / 10;
                    min++;
                    this.peakBAC();

            }

      return true;

    }

}
```

Stomach.java

```
package metab;
import metab.*;
import java.io.*;


/**
 * The <code>Stomach</code> class simulates the Stomach compartment in the
 * Compartmental model.  The stomach's job is two-fold in that it both metabolizes
 * and delays the transport of ethanol through the body.  Gastric Alcohol
 * Dehydrogenase (GADH) is responsible for the metabolism of some of the ethanol present
 * in the stomach.  At the same time, ethanol is exiting the stomach through the
 * pyloric sphincter at a rate determined by the current ethanol concentration
 * in the stomach.  The ethanol being metabolized should be removed from the system
```

```
 * by the user, just as in the <code>LeanBodyMass</code> class.  Ethanol exiting
 * through the pyloric sphincter should be transfered to an <code>Intestine</code>
 * class by the user.  Transfer and removal of ethanol and liquid should be handled
 * by the user using the addLiquid and addContents methods of the class.  To implement,
 * the amount of ethanol (g) and the total liquid voluem (L) must be present.  An optional
 * description (label) can also be used to better differentiate the compartment.
 * @author Levi Blackstone, Matthew Woller
 * @version 1.40
 * @see metab.Intestine
 * @see metab.LeanBodyMass
 * @see metab.NMSubject
 */
public class Stomach implements Serializable{

    /**
     * Amount of ethanol (grams).
     */
    private double amount;
    /**
     * Amount of total liquid in the stomach (L).
     */
    private double liquid;
      /**
     * Maximum reaction rate for Gastric Alcohol Dehydrogenase (GADH) in
     * mmol ethanol/min/L mucosa.
     */
    private static final double Vmax = 1.544;// 1.544 orig
    /**
     * Michaelis-Menton constant for GADH (mM).
     */
    private static final double Km = 497.834;// 497.834 orig
    /**
     * Ethanol transport rate constant from stomach to intestine (L/min).
     */
    private static final double ks = 0.083333;
    /**
     * Optional label for compartment.
     */
    private String label;

      /**
       * Constructor for stomach compartment.
       * @param amount Amount of ethanol (g).
       * @param liquid Amount of liquid including beverages and gastric fluid (L).
       * @see metab.Stomach#Stomach(double , double , String)
       */
    Stomach(double amount, double liquid) {

        this.amount = amount;
        this.liquid = liquid;

    }

      /**
       * Constructor for stomach compartment.
       * @param amount Amount of ethanol (g).
       * @param liquid Amount of liquid including beverages and gastric fluid (L).
```

```
   * @param label Optional description for compartment.
   * @see metab.Stomach#Stomach(double , double)
   */
Stomach(double amount, double liquid, String label) {

    this.amount = amount;
    this.liquid = liquid;
    this.label = label;

}

  /**
   * Adds ethanol to the intestine compartment (can be a negative value).
   * @param newContents Amount of ethanol being transfered (g).
   */
public void addContents(double newContents) {

    if (amount + newContents < 0.00) {
        amount = 0.00;
    } else {
              amount += newContents;
        }

}

  /**
   * Adds liquid from the beverage(s) and gastric fluid.
   * @param newLiquid Amount of liquid being transfered (L).
   */
public void addLiquid(double newLiquid){

  if (this.liquid + newLiquid < 0.00) {
        this.liquid = 0.00;
  } else {
              this.liquid += newLiquid; // L
        }

}

  /**
   * Resets compartment.
   */
public void clear() {

    amount = 0.0;

}

  /**
   * Computes amount of ethanol metabolized in stomach by GADH.
   * @return <Code>double</Code> representing amount of metabolized ethanol (mM).
   */
public double computeMMMetab() {

    if (liquid <= 0.0){
        return 0;
    } else {
```

```
        double concentration = amount/liquid; //g/L
        return (Vmax * concentration)/(Km + concentration); // mM
    }

}

  /**
   * Computes amount of ethanol being released into intestine.
   * @return <Code>double</Code> representing rate of release (g/min).
   */
public double computeEthanolRelease() {

    // Emptying rate continues to slow as ethanol decreases, but this
    // prevents the model from running indefinitely.
    if (amount < .6) {
        return .05;
    } else {
        if (liquid <= 0.00) {
            return .25;
        }
        double concentration = amount/liquid; //g/L
        return (ks * concentration); // g/min
    }
  }

  /**
   * Retrieves amount of ethanol remaining in intestine.
   * @return <Code>double</Code> representing amount of ethanol (g).
   */
public double getContents() {

    return amount;

}

  /**
   * Retrieves the optional label.
   * @return <Code>String</Code> representing label.
   */
public String getLabel() {

    return label;

}

  /**
   * Retrieves amount of liquid remaining in intestine.
   * @return <Code>double</Code> representing amount of liquid (L).
   */
public double getLiquid() {

  return liquid;

}

  /**
   * Checks to see if any ethanol remains in intestine.
```

```
       * @return <Code>boolean</Code> specifying if intestine is empty.
       */
    public boolean isEmpty() {

        return (amount <= 0.0);

    }

      /**
       * Sets optional label.
       * @param newLabel String representing new label.
       */
    public void setLabel(String newLabel) {

        label = newLabel;

    }

}
```

Intestine.java

```
package metab;
import metab.*;
import java.io.*;


/**
 * The <code>Intestine</code> class simulates the Small Intestine compartment
 * in the Compartmental model.  The only actual job that the small intestine
 * accomplishes in relation to the metabolism of ethanol is transport the ethanol
 * from the small intestine to the blood via the villi contained in the intestinal
 * wall.  No ethanol is actually metabolized in the small intestine, but it is
 * instead a factor that just delays transport to the <code>LeanBodyMass</code>.
 * The user must handle the transport of the ethanol and liquid between other
 * compartments by using the addLiquid and addContents methods for each compartment.
 * To implement, all that is required is the amount of ethanol (g), and the total
 * liquid volume (L).  An optional extra description (label) may be included, but
 * is not required for the class to function properly.
 * @author Levi Blackstone, Matthew Woller
 * @version 1.19
 * @see metab.Stomach
 * @see metab.LeanBodyMass
 * @see metab.NMSubject
 */
public class Intestine implements Serializable{

    /**
     * Ethanol transport rate constant from intestine to lean body mass (L/min).
     */
    private static final double ki = 0.107;//.107
    /**
     * Amount of ethanol (g).
     */
    private double amount;
    /**
     * Amount of total liquid in the small intestine (L).
```

```
 */
private double liquid;
  /**
   * Optional label for Compartment.
   */
private String label;

  /**
   * Constructor for intestine compartment.
   * @param amount Amount of ethanol (g).
   * @param liquid Amount of liquid including beverages and gastric fluid (L).
   * @see metab.Intestine#Intestine(double , double , String)
   */
Intestine(double amount, double liquid) {

    this.amount = amount;
    this.liquid = liquid;

}

  /**
   * Constructor for intestine compartment.
   * @param amount Amount of ethanol (g).
   * @param liquid Amount of liquid including beverages and gastric fluid (L).
   * @param label Optional description for compartment.
   * @see metab.Intestine#Intestine(double , double)
   */
Intestine(double amount, double liquid, String label) {

    this.amount = amount;
    this.liquid = liquid;
        this.label = label;

}

  /**
   * Adds ethanol to the intestine compartment (can be a negative value).
   * @param newContents Amount of ethanol being transfered (g).
   */
public void addContents(double newContents) {

    if (amount + newContents < 0.00) {
        amount = 0.00;
    } else {
              amount += newContents;
        }

}

  /**
   * Adds liquid from the beverage(s) and gastric fluid.
   * @param newLiquid Amount of liquid being transfered (L).
   */
public void addLiquid(double newLiquid) {

  if (this.liquid + newLiquid < 0.00) {
        this.liquid = 0.00;
```

```
  } else {
              this.liquid += newLiquid; // L
        }

}

  /**
   * Resets compartment.
   */
public void clear() {

    amount = 0.0;
    liquid = 0.0;

}

  /**
   * Retrieves amount of ethanol remaining in intestine.
   * @return <Code>double</Code> representing amount of ethanol (g).
   */
public double getContents() {

    return amount;

}

  /**
   * Retrieves the optional label.
   * @return <Code>String</Code> representing label.
   */
public String getLabel() {

    return label;

}

  /**
   * Retrieves amount of liquid remaining in intestine.
   * @return <Code>double</Code> representing amount of liquid (L).
   */
public double getLiquid() {

  return liquid;

}

  /**
   * Checks to see if any ethanol remains in intestine.
   * @return <Code>boolean</Code> specifying if intestine is empty.
   */
public boolean isEmpty() {

    return (amount <= 0.0);

}

  /**
```

```
        * Computes amount of ethanol being absorbed into lean body mass.
        * @return <Code>double</Code> representing rate of absorption (g/min).
        */
    public double computeEthanolAbsorption() {

        if (liquid <= 0.0){
            return amount;
        } else {
            double concentration = amount/liquid; //g/L
            return (ki * concentration); // g/min
        }

      }

      /**
       * Sets optional label.
       * @param newLabel String representing new label.
       */
    public void setLabel(String newLabel) {

        label = newLabel;

    }

}
```

LeanBodyMass.java

```
package metab;
import metab.*;
import java.io.*;


/**
 * The <code>LeanBodyMass</code> class simulates the Lean Body Mass compartment
 * in the Compartmental model.  The Lean Body Mass handles the majority of the
 * actual metabolism of ethanol in the human body, mainly through the liver.
 * Once the ethanol has been metabolized, the user should remove it from the system
 * using the addContents method.  No addLiquid method is necessary for this class
 * because the liquid is the aqueous portion of the subject's body.  This Total Body
 * Water is a required argument during class construction.  The TBW must be calculated
 * or measured outside of the class (externally).  Also required during construction
 * is the amount of ethanol (g), the subject's previous drinking history (an integer
 * value consisting of one of the constants defined in <code>NMSubject</code>,
 * and, optionally, a description (label) of the compartment.
 * @author Levi Blackstone, Matthew Woller
 * @version 1.61
 * @see metab.Stomach
 * @see metab.Intestine
 * @see metab.NMSubject
 */
public class LeanBodyMass implements Serializable{

    /**
     * Amount of ethanol (grams).
     */
```

```
private double amount;
/**
 * Total Body Water (TBW) present in individual.
 */
private double TBW;
/**
 * Liver Alcohol Dehydrogenase (LADH) maximum reaction rate (mmol ethanol/min
 * /L Body Water).
 */
private static final double Vmax = 4.518;// 4.518 orig
/**
 * Michaelis-Menton constant for LADH (mM).
 */
private static final double Km = 1.752;// 1.752 orig
/**
 * Standard representation of the subject's drinking history.
 */
private int drinkingHistory;
/**
 * Optional label for compartment.
 */
private String label;

  /**
   * Constructor for lean body mass compartment.
   * @param amount Amount of ethanol (g).
   * @param TBW Total water in body mass (L).
   * @param drinkingHistory Previous drinking habits of the subject.
   * @see metab.LeanBodyMass#LeanBodyMass(double , double , int, String)
   */
LeanBodyMass(double amount, double TBW, int drinkingHistory) {

    this.amount = amount;
    this.TBW = TBW;
    this.drinkingHistory = drinkingHistory;

}

/**
   * Constructor for lean body mass compartment.
   * @param amount Amount of ethanol (g).
   * @param TBW Total water in body mass (L).
   * @param drinkingHistory Previous drinking habits of the subject.
   * @param label Optional description for compartment.
   * @see metab.LeanBodyMass#LeanBodyMass(double, double, int)
   */
LeanBodyMass(double amount, double TBW, int drinkingHistory, String label) {

    this.amount = amount;
    this.TBW = TBW;
    this.drinkingHistory = drinkingHistory;
    this.label = label;

}

  /**
   * Adds ethanol to the intestine compartment (can be a negative value).
```

```
    * @param newContents Amount of ethanol being transfered (g).
    */
public void addContents(double newContents) {

    if (amount + newContents < 0.00) {
        amount = 0.00;
    } else {
              amount += newContents;
        }

}

  /**
    * Resets compartment.
    */
public void clear() {

    amount = 0.0;

}

  /**
    * Computes amount of ethanol metabolized in lean body mass by LADH.
    * @return <Code>double</Code> representing amount of metabolized ethanol (mM).
    */
public double computeMMMetab() {

    // Accelerates, retards, or leaves alone the rate of metabolism.
    double concentration = amount/TBW;
    return (((drinkingHistory == NMSubject._LIGHT_DRINKER) ? Vmax * .8 :
             (drinkingHistory == NMSubject._MODERATE_DRINKER) ? Vmax :
             (drinkingHistory == NMSubject._HEAVY_DRINKER) ? Vmax * 1.2 : Vmax)
              * concentration) / (Km + concentration);

}

  /**
    * Retrieves amount of ethanol remaining in intestine.
    * @return <Code>double</Code> representing amount of ethanol (g).
    */
public double getContents() {

    return amount;

}

  /**
    * Retrieves the optional label.
    * @return <Code>String</Code> representing label.
    */
public String getLabel() {

    return label;

}

  /**
```

```
        * Checks to see if any ethanol remains in intestine.
        * @return <Code>boolean</Code> specifying if intestine is empty.
        */
    public boolean isEmpty() {

        return (amount <= 0.0);

    }

      /**
        * Sets optional label.
        * @param newLabel String representing new label.
        */
    public void setLabel(String newLabel) {

        label = newLabel;

    }

}
```

Drink.java

```
package metab;
import java.io.*;


/**
 * The <code>Drink</code> class is a simple way to represent an alcoholic
 * beverage.  Values used include the ethanol concentration (%), the volume (oz),
 * and the name of the beverage.  The percent volume of ethanol can be determined
 * by using the total volume and the concentration.  This class allows for infinite
 * numbers of different drinks.
 * @author Levi Blackstone, Matthew Woller
 * @version 1.03
 */
public class Drink implements Serializable{

      /**
        * Concentration of beverage (percent).
        */
      private double concentration;
      /**
        * Volume of beverage (oz).
        */
      private double volume;
      /**
        * Name of the beverage.
        */
      private String name;

      /**
        * Constructor for Drink class.
        * @param concentration Concentration of beverage (percent).
        * @param volume Volume of beverage (oz).
        */
```

```
	Drink(double concentration, double volume, String name) {

		this.concentration = concentration;
		this.volume = volume;
		this.name = name;

	}

	/**
	 * Retrieves concentration of beverage.
	 * @return <Code>double</Code> representing concentration of beverage (percent).
	 */
	public double getConcentration() {

		return concentration;

	}

	/**
	 * Retrieves the name and description of the drink.
	 * @return <code>String</code> containing the drink name.
	 */
	public String getName() {

		return this.name;

	}

	/**
	 * Retrieves volume of beverage.
	 * @return <Code>double</Code> representing volume of beverage (oz).
	 */
	public double getVolume() {

		return volume;

	}

	/**
	 * Retrieves a copy of beverage concentration and volume (percent, L).
	 * @return <Code>Drink</Code> representing new instance of current beverage.
	 */
	public Drink getCopy() {

		return new Drink(this.concentration, this.volume, this.name);

	}

}
```

DrinkTime.java

```
package metab;
import java.io.*;

/**
```

```
 * The <code>DrinkTime</code> class has use only in containing a time of day.
 * It does not specify a date for simplicity.  It relies on hours, minutes, and
 * a 12-hour (non-military) clock to hold the time.  This class was designed
 * instead of using the <code>Date</code>-types because neither milliseconds nor
 * specific dates were required.
 * @author Levi Blackstone, Matthew Woller
 * @version 1.03
 */
public class DrinkTime implements Serializable{

	/**
	 * Standard value representing AM hours.
	 */
	public static final int _AM = 1;
	/**
	 * Standard value representing PM hours.
	 */
	public static final int _PM = 0;
	/**
	 * Number of hours.
	 */
	private int hours;
	/**
	 * Number of minutes.
	 */
	private int minutes;
	/**
	 * Specifies whether time is AM or PM.
	 */
	private int AMorPM;

	/**
	 * Constructor for DrinkTime class.
	 * @param hours Number of hours.
	 * @param minutes Number of minutes.
	 * @param AMorPM Specifies whether time is AM or PM.
	 */
	DrinkTime(int hours, int minutes, int AMorPM) {

		this.hours = hours;
		this.minutes = minutes;
		this.AMorPM = AMorPM;

	}

	/**
	 * Retrieves number of hours.
	 * @return <Code>int</Code> representing number of hours.
	 */
	public int getHours() {

		return hours;

	}

	/**
	 * Retrieves number of minutes.
```

```
 * @return <Code>int</Code> representing number of minutes.
 */
public int getMinutes() {

	return minutes;

}

/**
 * Retrieves 12-hour format information.
 * @return <Code>int</Code> representing whether time is AM or PM.
 */
public int getAMorPM() {

	return AMorPM;

}

/**
 * Retrieves a copy of time encompassing drinking period.
 * @return <Code>DrinkTime</Code> representing new instance of current time.
 */
public DrinkTime getCopy() {

	return new DrinkTime(hours, minutes, AMorPM);

}

/**
 * Sets number of hours.
 * @param hours Number of hours.
 */
public void setHours(int hours) {

	this.hours = hours;

}

/**
 * Sets number of minutes.
 * @param minutes Number of minutes.
 */
public void setMinutes(int minutes) {

	this.minutes = minutes;

}

/**
 * Sets whether time is AM or PM.
 * @param AMorPM Specifies whether time is AM or PM.
 */
public void setAMorPM(int AMorPM) {

	this.AMorPM = AMorPM;

}
```

```
}
```

TDSubject.java

```
package metab;
import java.io.*;

public class TDSubject implements Serializable{

	public static final int _LIGHT = 1;
	public static final int _MODERATE = 1 << 1;
	public static final int _HEAVY = 1 << 2;
	public static final int _BEER = 1 << 3;
	public static final int _WINE = 1 << 4;
	public static final int _HARD_LIQUOR = 1 << 5;
	public static final int _METRIC = 1 << 6;
	public static final int _IMPERIAL = 1 << 7;
	public static final int _ERROR = 1 << 8;
	private int bitFlag, age;
	private double numberOfDrinks, hoursDrinking, weight;
	private char gender;
	private String subjectName;


	TDSubject(int age, char gender, double numberOfDrinks, double hoursDrinking,
			String subjectName, double weight, int bitFlag){

		this.age = age;
		this.gender = gender;
		this.numberOfDrinks = numberOfDrinks;
		this.hoursDrinking = hoursDrinking;
		this.subjectName = subjectName;
		this.bitFlag = bitFlag;
		if ((bitFlag & _METRIC) != 0)
			this.weight = weight;
		else if ((bitFlag & _IMPERIAL) != 0)
			this.weight = weight / 2.20462262; // lb to kg

	}

	public int getAge(){

		return age;

	}

	public double getBAC(){

		double oz = 0, BAC, hourlyDecrease = 0;
	  double TBW = ((gender == 'M')?(weight * .58):(weight * .49)) * 1000;
			//Total Body Water in milliliters
	  double gramsIn100mlBlood = (23.36 / TBW) * .806 * 100;
			//Grams alcohol per one hundred milliliters blood
		if ((bitFlag & _BEER) != 0)
		oz = numberOfDrinks * 12 * 0.06;
```

```
    else if ((bitFlag & _WINE) != 0)
        oz = numberOfDrinks * 10 * 0.10;
    else if ((bitFlag & _HARD_LIQUOR) != 0)
        oz = numberOfDrinks * 1 * 0.48;

    //Actual amount of ethanol consumed in ounces
    if ((bitFlag & _LIGHT) != 0)
        hourlyDecrease = .012;
    else if ((bitFlag & _MODERATE) != 0)
        hourlyDecrease = .017;
    else if ((bitFlag & _HEAVY) != 0)
        hourlyDecrease = .020;
    //Hourly decrease in BAC
        BAC = oz * gramsIn100mlBlood;//Instantaneous BAC
    BAC = BAC - (hoursDrinking * hourlyDecrease);//Final BAC for the subject
    return BAC;

}

  public int getDrinkingHistory(){

        if ((bitFlag & _LIGHT) != 0)
              return _LIGHT;
        else if ((bitFlag & _MODERATE) != 0)
              return _MODERATE;
        else if ((bitFlag & _HEAVY) != 0)
              return _HEAVY;
        else
              return _ERROR;

}

public char getGender(){

  return gender;

}

public double getNumberOfDrinks(){

        return numberOfDrinks;

}

  public double getHoursDrinking(){

        return hoursDrinking;

  }

  public String getSubjectName(){

        return subjectName;

  }

  public int getTypeOfDrink(){
```

```
            if ((bitFlag & _BEER) != 0)
                  return _BEER;
            else if ((bitFlag & _WINE) != 0)
                  return _WINE;
            else if ((bitFlag & _HARD_LIQUOR) != 0)
                  return _HARD_LIQUOR;
            else
                  return _ERROR;

      }

      public double getWeight(){

            if ((bitFlag & _IMPERIAL) != 0){
            return weight * 2.20462262;
        }
        else {
            return weight;
        }

      }

}
```

Driver.java

```
package metab;
import metab.*;
import java.io.*;
import java.text.*;

class Driver {

    private static NumberFormat setp = new DecimalFormat("000.000");

    public static void get_compute_store(String infile, String outfile) throws IOException,
FileNotFoundException {// for use only with orignal test data -

// modify appropriately if needs change
        BufferedReader in = new BufferedReader( new FileReader( infile ) );//input
filestream
        BufferedWriter out = new BufferedWriter( new FileWriter( outfile ) );//output
filestream
        String inputbuffer;
        int numofinputsubsmen = 0, numofinputsubswomen = 0;
        double difference_TBW_M, difference_BAC_M, difference_TBW_F, difference_BAC_F;
        double sigma_TBW_M = 0;//
        double sigma_TBW_F = 0;//          These are used only for our new model
        double sigma_BAC_F = 0;//          Other purposes could be designed if modified
        double sigma_BAC_M = 0;//
        out.write("(sex    obs. BAC   obs. TBW   pred. BAC    pred. TBW)");
        out.newLine();
        out.newLine();
        while ((inputbuffer = in.readLine()) != null){
```

```
            char sex_MF = in.readLine().charAt(0);
            int age_in_yrs = Integer.parseInt(in.readLine());
            double h_in_cm = Double.parseDouble(in.readLine());
            double w_in_kg = Double.parseDouble(in.readLine());
            double observed_BAC = Double.parseDouble(in.readLine());
            double observed_TBW = Double.parseDouble(in.readLine());
            Drink [] thedrinks = {new Drink(20,.2 / 0.0295735297, "200 mL of 20%
alcohol")};
            NMSubject sub = new NMSubject(age_in_yrs, h_in_cm, sex_MF, w_in_kg, thedrinks,
                        new DrinkTime(12,0,DrinkTime._PM), new
DrinkTime(2,0,DrinkTime._PM),
                        "Boba", NMSubject._NO_MEAL | NMSubject._MODERATE_DRINKER |
NMSubject._METRIC);
            sub.runModel();
            if (sub.getGender() == 'M'){
                numofinputsubsmen++;
                difference_TBW_M = observed_TBW - sub.getTBW();
                difference_BAC_M = observed_BAC - sub.getPeakBAC();
                sigma_TBW_M += Math.pow(difference_TBW_M, 2);
                sigma_BAC_M += Math.pow(difference_BAC_M, 2);
            }
            else if (sub.getGender() == 'F'){
                numofinputsubswomen++;
                difference_TBW_F = observed_TBW - sub.getTBW();
                difference_BAC_F = observed_BAC - sub.getPeakBAC();
                sigma_TBW_F += Math.pow(difference_TBW_F, 2);
                sigma_BAC_F += Math.pow(difference_BAC_F, 2);
            }
            out.write( "  " + sex_MF + "     " + setp.format(observed_BAC) + "     "
                        + setp.format(observed_TBW) + "     " + setp.format(sub.getPeakBAC())
+ "     " + setp.format(sub.getTBW()));//output to file stream
            out.newLine();
        }
        double rms_TBW_M = Math.sqrt(sigma_TBW_M / numofinputsubsmen);
        double rms_BAC_M = Math.sqrt(sigma_BAC_M / numofinputsubsmen);
        double rms_TBW_F = Math.sqrt(sigma_TBW_F / numofinputsubswomen);
        double rms_BAC_F = Math.sqrt(sigma_BAC_F / numofinputsubswomen);
        out.write("rms TBW men: " + setp.format(rms_TBW_M) + "  rms TBW women: "
                  + setp.format(rms_TBW_F));
        out.newLine();
        out.write("rms BAC men: " + setp.format(rms_BAC_M) + "  rms BAC women: " +
setp.format(rms_BAC_F));//ouput to file stream
        out.close();//close file stream
        in.close();//close file stream

    }

      public static void main(String args[]) throws IOException{
        get_compute_store("inputdata.txt", "outputdata.txt");
      }

}
```