Sample CS100B: Prelim 1
Date
Time

(Print Your Name)

(Signature)

(Section)

(Student ID)

Instructions:

- Answer all questions by yourself!
- Sign or initial each page.
- Show all work and comment code fragments to receive partial credit
- Use the back of each page if you need more space.
- Remember to breathe! Relax: it's only a test.

Points:

1. _____ **(10 points)**
2. _____ **(10 points)**
3. _____ **(20 points)**
4. _____ **(25 points)**
5. _____ **(35 points)**

Subtotal: _____**/100 points**

6. _____ **(2 possible bonus points)**

Total: _____

Problem 1 *Math to Java* (10 points)

Convert the following mathematical statements into Java code fragments that perform the same operations and functions. Do not manipulate the expression before converting it into Java. However, you are welcome to use as many parentheses as you wish. Assume that all variables *a*, *b*, and *c* contain floating-point values and are non-zero.

Example) $\frac{a}{2}$

Java code: `a/2`

1a) $\left(\frac{a}{b}\right)^{\frac{2}{3}}$ (2 points)

1b) $\sin(123°)$ (2 points)

1c) $|a + b|$ (3 points)

1d) $\frac{3}{4} + \frac{a}{b + \frac{2}{b+2}}$ (3 points)

Problem 2 *Spot the Bugs!* (10 points)

The following program written in Java is producing many compiler and run-time errors for the developer. Assume that your Java compiler interface accesses **`TokenReader.java`**. Indicate the mistakes in the code by either circling the incorrect portions or writing code that might be missing. You do not need to correct the code that you circle. Hint: There are many mistakes present: some code to be circled, some code to be added.

```
 * bug ridden program! */

/  Please fix me.

public CLASS Sum_for {

     pubic static void main(String args[]) {

          TokenReader in = new TokenReader(System_in);

          int    count  = 0

          integer stop  = 0

          DOUBLE sum    = 0

          double temp   = 0

          System.out.println('How many numbers do you wish to enter? ');

          stop assigns in.readInt();

          for(count == 1; count =< stop; count++) {

               System  out.println("Enter data #" + count + ": ");

               temp1 = in.readDouble();

               temp + sum = sum;

          System.out.Println();

          System.out.println("Final sum: "   sum);

     ]

)
```

Problem 3 *Tracing* (20 points)

What is the output of the following code? Hint: Yes, this code does in fact compile and run.

```
public class Find_do_while {
     public static void main(String args[]) {

          int a = 0;
          int b = 0;
          int max = 0;
          int min = 0;

          do {
               a = -a*2;
               b = -(b + a++);
               System.out.println("a:" + a + "b:" + b);

               if(b-a > max)
                    max = b-a;

          } while( a + b > -1);

          System.out.println("Final values");
          System.out.println("a: " + a + "b: " + b);
          System.out.println("Max b-a: " + max);
     }
}
```

Problem 4 *Data for Plot* (25 points)

You wish to generate data that will help you plot the equation $y = f(x) = 3x - x^3$ from $0 \le x \le 2$. Finish writing the following program that will print data that can be used to plot this equation. Choose an increment size of 0.1 for x. After printing the data, the program should report the minimum value of $f(x)$ found during the iteration by the program. (Do not use a derivative to find the minimum! Let the code do the work.) Hint: The program should print x and y values in separate columns separated by one space. Print the minimum value of $f(x)$ after the x and y values.

NOTE: Fill in the blanks wherever you see a line, like _____, including comments!

```
public class plot_sample {
     public static void main(String args[]) {

     double x;
     double y=0;

     double start = _______ // starting value of ____

     double stop  = _______ // stopping value of ____

     double inc = _______ // increment for x

     double min;

     x = start;

     min = ___________________

     while( ____________ ) {

          y = Math.pow(x,3) - 3*x;

          if(y _______ min)

               ________________;

          x = ________________

          System.out.println(_________ + _________ + ________);
     }

     System.out.println(_________);
     }
}
```

Problem 5 *Newton's Method* (35 points: 25 for code, 10 for documenting)

Write a program that uses Newton's Method to solve for the real positive root of the equation $x^4 - 7x - 60 = 0$. Your root should satisfy the equation within a tolerance $\varepsilon = 0.001$ of the true root, *not* the y value. So, check the tolerance with the root! YOU MUST PROVIDE COMMENTARY!

Hints:

- Start your program with the code and variables provided below.
- Use an initial value of **2** for the root. Also, $f'(x) = 4x^3 - 7$.
- Recall that Newton's method solves for roots with the formula $x_{new} = x_{old} - \frac{f(x_{old})}{f'(x_{old})}$.

```
//provided code fragment for root solving with Newton's Method

public class Newton {
     public static void main(String args[]) {

     // use these variables!
     double root   = 2;      //the initial value of the root
     double EPS    = 0.001;  //the tolerance value for error
     double y;
     double yp;

// now finish this program!
```

Problem 6 (2 points) *Bonus problems (optional!)*

6a) What does <SOMETHING> stand for? (0.5 point)

6b) Who <SOMETHING> the term <SOMETHING> ? (0.5 point)

6c) Why might <SOMETHIING> be considered "<SOMETHING>" (0.5 point)

6d) What does a <SOMETHING> do? (0.5 point)