CS100B: Prelim 1 SOLUTIONS
September 27, 1999
7:30 PM – 9:00 PM

____
(Print your name)

____
(Signature)

____
(Student ID)

Sections	10	11	12	13	14	15
(circle one)	Mon	Mon	Mon	Tue	Tue	Tue
	1:25	2:30	3:35	10:10	2:30	3:35

Instructions:
- Answer all questions by yourself! Respect academic integrity.
- 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. Do not declare any additional variables.

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

Java code: **`a/2`**

1a) $\frac{ab}{\sqrt{c}}$ (2 points)

`a*b/Math.pow(c,0.5)` or **`a*b/Math.sqrt(c)`**

1b) $\tan(37°)$ (2 points)

`Math.tan(Math.PI*37/180)`

1c) $\frac{a+b}{2}|c|$ (3 points)

`Math.abs(c)*(a+b)/2`

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

`a + 2.0/(3 + 4/b)`

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 circling the incorrect portions. You do not need to correct the code that you circle. Hint: There are 10 mistakes present.

```
/   bug-ridden program: please fix me!

pubic class Project1 {

     public static void MAIN(String args[]) {

        integer k;

          // initialize Text object in to read from standard input
          TokenMaster in = new TokenReader(System.in);

          // Read and write some numbers and strings
          System.out.println("Please enter an integer: ");

          // Read integer
          k = in.readInt()

          // Perform algorithm
          while ( k > 1 )

               if ( (k % 2) = 0) {       // test if k is even

                    k / 2 = k;
                    System.out.println(k + " after dividing by 2");

               }

               else; {                   // k is odd

                    k = 3 * k + 1;
                    System.out.println(k + " after multiplying by 3 " +
                         "and adding 1");

               }

          System.out.println("Done!");

        }

     }

}
```

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 Repeat_do_while_test {

     public static void main(String args[]) {

     int count = 1;

     do {
          System.out.println(count);

          if(count==0)
               System.out.println("zero");
          else if (count % 2 == 0)
               System.out.println("even number");
          else if(count > 0)
               System.out.println("odd positive");
          else if(count < 0)
               System.out.println("odd negative");
          else
               System.out.println("What's that?");
          count--;

          } while(count > -3);

          System.out.println(count);

     }
}
```

Output:

```
1
odd positive
0
zero
-1
odd negative
-2
even number
-3
```

Problem 4 *Data for Plot* (25 points)

You wish to generate data that will help you plot the equation $y = f(x) = 4 - x^2$, given $-2 \le x \le 2$. Finish writing the following program that will print data that can be used to plot this equation. Choose an increment size of 0.2 for x. After printing the data, the program should report the maximum value of $f(x)$ found during the iteration by the program. (Do not use calculus to find the maximum! Let the code do the work.) Hint: The program should print x and y values in separate columns separated by one space. Print the maximum value of $f(x)$ after finishing the output of all the x and y values. NOTE: Fill in the blanks wherever you see a line, like _______, including comments!

```
public class plot {

     public static void main(String args[]) {

     double x = 0;

     double y = 0;

     double start = -2 ;        // starting value of x

     double stop  = 2 ;         // stopping value of x

     double inc   = 0.2 ;       // increment of x

     double x_at_ymax = start;  // initial value of x_at_ymax is "start".

     // use x_at_ymax to compute initial ymax:

     double ymax = -(x_at_ymax*x_at_ymax) + 4;

     for( x = start ; x<=stop ; x=x+inc ){

          y = -x*x+4;

          System.out.println( x + " " + y );

          if( y > ymax ) {

               ymax = y ;

               x_at_ymax = x ;

          }
     }

     System.out.println("Max value of y = " + ymax );

     System.out.println("Occurs at x = " + x_at_ymax );
     }
}
```

was typo original

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 negtaive root of the equation $x^3 - 2x + 4 = 0$. Rather than checking for tolerance of the root, your program should check the tolerance for the function $f(x)$, and NOT the root. Your program must also count the number of iterations performed during the analysis. Use a tolerance of $\varepsilon = 0.001$. You must output the final value of the root and the number of iterations. NOTE: YOU MUST PROVIDE COMMENTS FOR YOUR CODE! (25 points for code; 10 points for documentation)

Hints:

- Start your program with the code and variables provided below.
- Use an initial value of **1** for the root. Also, $f'(x) = 3x^2 - 2$.
- Recall that Newton's method solves $x_{new} = x_{old} - \frac{f(x_{old})}{f'(x_{old})}$.

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

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

        // Use these variables! Declare other variables you might need.
        double root   = 1;      //the initial value of the root
        double EPS    = 0.001;  //the tolerance value for error
        int iteration = 0;      //counter for iterations

        double y;               //y=f(x)
        double yp;              //y'=f'(x)

        // Iterate for root iterate until tolerance of y is met
        do {
            ++iteration;
            y =  Math.pow(root,3)-2*root+4;    // compute f(x) for current x
            yp = 3*Math.pow(root,2)-2;         // compute f'(x) for current x
            root = root - y/yp;                // increment root
        } while (Math.abs(y) >= EPS);          // check tolerance of y value

        // could technically now say root=root+yp, but not necessary
        // the following line is not required (checks final result)
        y =  Math.pow(root,3)-2*root+4;

        // required output
        System.out.println("x: " + root);
        System.out.println("y: " + y);

        //print iterations
        System.out.println("iterations: " + iteration);

    }
}
```

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

6a) What does "OOP" stand for (in the context of Java programming)? (0.5 point)

Object Oriented Programming

6b) Who named the term "LHS/RHS" for the LHS/RHS method? (0.5 point)

David I. Schwartz
Schwartz
Professor Schwartz
Dr. Schwartz
our professor
(you get the idea :-)

6c) Why might **`break`** or **`goto`** statements be considered "evil?" (0.5 point)

Makes code hard to follow.

6d) What does a sheet pile do? (0.5 point)

Shores up soil.