CS100B: Prelim 2 Sample
October 19, 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:
• CIRCLE YOUR SECTION! Otherwise, your test will be sent to Carpenter.
• 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. _____ (15 points)

3. _____ (20 points)

4. _____ (25 points)

5. _____ (30 points)

Subtotal: _____/100 points

6. _____ (2 possible bonus points)

Total: _____
**Problem 1**  (10 points) Short Answer questions

Answer the following questions. Try to keep your answers concise. (Know the definitions to all the following terms to answer the 5 questions I’m not showing.) Example: Describe the client-server relationship. (Keep answers brief!)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstraction</td>
<td></td>
</tr>
<tr>
<td>actual parameter</td>
<td>object</td>
</tr>
<tr>
<td>address</td>
<td>object oriented programming</td>
</tr>
<tr>
<td>alias</td>
<td>parameter</td>
</tr>
<tr>
<td>attribute</td>
<td>pass by reference</td>
</tr>
<tr>
<td>behavior</td>
<td>pass by value</td>
</tr>
<tr>
<td>call</td>
<td><strong>private</strong></td>
</tr>
<tr>
<td>class</td>
<td>procedural programming</td>
</tr>
<tr>
<td>class variable</td>
<td><strong>public</strong></td>
</tr>
<tr>
<td>class</td>
<td>reference</td>
</tr>
<tr>
<td>client</td>
<td>reference variable</td>
</tr>
<tr>
<td>constructor</td>
<td>return type</td>
</tr>
<tr>
<td>final</td>
<td>return value</td>
</tr>
<tr>
<td>flow of control</td>
<td>server</td>
</tr>
<tr>
<td>formal parameter</td>
<td>service method</td>
</tr>
<tr>
<td>garbage collection</td>
<td>state</td>
</tr>
<tr>
<td>identity</td>
<td><strong>static</strong></td>
</tr>
<tr>
<td>information hiding</td>
<td>static variable</td>
</tr>
<tr>
<td>instance variable</td>
<td>string</td>
</tr>
<tr>
<td>instantiation</td>
<td><strong>String</strong></td>
</tr>
<tr>
<td>interval analysis</td>
<td>structured programming</td>
</tr>
<tr>
<td>invoke</td>
<td>support method</td>
</tr>
<tr>
<td>Java</td>
<td>type</td>
</tr>
<tr>
<td>local variable</td>
<td>visibility</td>
</tr>
<tr>
<td>method</td>
<td>visibility modifier</td>
</tr>
<tr>
<td>method definition</td>
<td><strong>void</strong></td>
</tr>
<tr>
<td>modifier</td>
<td></td>
</tr>
<tr>
<td>method invocation</td>
<td></td>
</tr>
<tr>
<td>method invocation</td>
<td></td>
</tr>
</tbody>
</table>
Problem 2  (15 points) Tracing

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

```java
// Class for an integer Number
class Num{
    int value;
    Num() {}  
    public void add(Num number){
        number.value++;
    }
}

// Demonstrates the effects of parameter passing
class Parameter_Passing{
    public static void print(int value1, Num value2){
        System.out.println("value1= "+value1);
        System.out.println("value2= "+value2.value);
        System.out.println();
    } // method print

    public static void change1(int value1, Num value2){
        value1++;
        value2.value++;
        print(value1,value2);   // 2nd Output
    } // method change1

    public static int change2(int value1, Num value2){
        value1++;
        value2.value++;
        print(value1,value2);   // 4th Output
        return value1;
    } // method change2

    public static void main (String[] args){
        int number1=5;
        Num number2=new Num();
        number2.value=10;
        print(number1,number2); // 1st Output

        change1(number1, number2);
        print(number1,number2); // 3rd Output

        number1=change2(number1, number2);
        print(number1,number2); // 5th Output

        number2.add(number2);
        print(number1,number2); // 6th Output
    } // method main
} // class Parameter_Passing
```
Problem 3  (20 points) Nested Loops

Write a program that will generate the following tabular output for a user entered integer n. The main “diagonal” (the elements whose rows and columns match) must values of 1. All other elements are zero. See example output for different values of n.

\[ n = 1 \rightarrow 1 \]

\[ n = 2 \rightarrow \begin{array}{cc}
1 & 0 \\
0 & 1 \\
\end{array} \]

\[ n = 3 \rightarrow \begin{array}{ccc}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array} \]

Your output won’t have the \( n = \# \rightarrow \) portion, just the table of 0s and 1s. Though not required, documenting your code will help you earn partial credit

```java
public class Identity {
    public class static void main(String args[]) {
        // complete the code
    }
}
```
Problem 4 (25 points) Complete the class!

We're already familiar with interval arithmetic for adding two intervals. Now we wish to extend this notion to add three intervals. Using the code below as a starting point, write a method called `addInterval` which can be invoked on an `Interval` object. This method should take two `Interval` objects as arguments and return the sum of the two `Interval` objects and the `Interval` object that invoked the addition method.

Note: Though not required, documenting your code will earn partial credit.

```java
public class Interval {

    public double minimum;
    public double maximum;

    public Interval(double min, double max) {
        minimum = min;
        maximum = max;
    }

    // complete the addition code here
```
Problem 5 (30 points) Program (20 points correctness, 5 points style, 5 points documentation: all mandatory!)

Write a program that performs interval subtraction and division between two intervals. Assume that a user enters the lower and upper bounds of both intervals. The user must also choose the operation. You should use a class that called Interval that contains all appropriate methods and variables. You do not need to employ encapsulation. Be sure to check if the user enters a zero and chooses division.
Problem 6  (2 points) Bonus problems (optional! REGRADE REQUESTS UNLIKELY TO BE ACCEPTED)

6a) (0.5 points) What does <something> stand for (in <something>)?

6b) (0.5 points) Describe <somethings> that guarantee <something> of <something>.

6c) (0.1 points) Who <did something> to <something> to <do something>?

6d) (0.1 points) What does the expression <something> mean?

6e) (0.1 points) What is the <something> given <something> and <something>?

6f) (0.1 points) What does <something> do?

6g) (0.1 points) What does <a different something> do?

6h) (0.5 point) What is the <something> of <someone's> <something>?