CS100B: Prelim 3 Solutions
November 16, 1999
7:30 PM – 9:00 PM

____

(Print your name)

Statement of integrity: I did not break the rules of academic integrity on this exam:

____

(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 goes to Carpenter.
- Answer all questions by yourself! Respect academic integrity.
- This test is closed book – no calculators, reference sheets, or any other material allowed.
- Initial or sign 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. ______ (15 points)

2. ______ (25 points)

3. ______ (35 points)

4. ______ (25 points)

Subtotal: ______ /100 points

5. ______ (2 possible bonus points)

Total: ______

Problem 1 *(15 points) Trace the output.*

Fill in the output for the following code (5 points each). Note that some portions are provided for you. Show your work by drawing pictures and/or describing your steps for partial credit.

```
public class problem1 {
   public static void main(String args[]) {
      Data0 d0 = new Data0(0);
      Data1 d1 = new Data1(1,2);
      d0.D0 = new Data0(3);
      d1.D0 = new Data1(4,5);
      d1.D0.D0 = new Data0(6);
      System.out.println("1: " + d0.get_val() );
      System.out.println("2: " + d1.get_val() );
      System.out.println("3: " + d1.D0.get_val() );
   } // method main
} // class problem2

class Data0 {
   private int m;
   public Data0 D0;
   Data0(int input) {
      m = input;
   }
   protected int get_val() {
      return m+D0.m;
   }
} // class Data0

class Data1 extends Data0 {
   private int n;
   Data1(int x, int y) {
      super(x);
      n = y;
   }
   protected int get_val() {
      return n+super.get_val();
   }
} // class Data1
```

Output:

```
1: 3

2: 7

3: 15
```

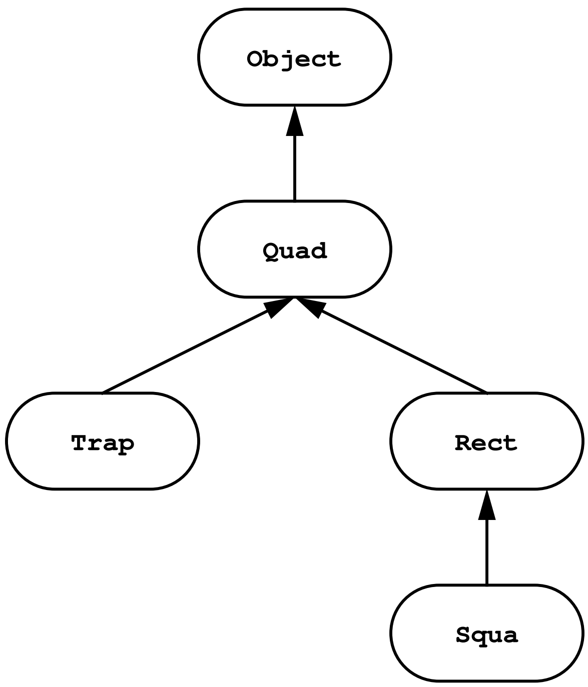
Problem 2 *25 points: Software development and programming. Both parts (a) and (b) are mandatory.*

2a) (3 points)

Suppose you need to develop a program that computes perimeter for different kinds of quadrilaterals using objects. From your study of geometry you know the following relationships:

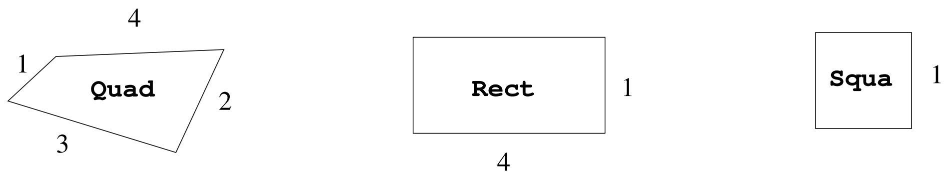
- A trapezoid is a quadrilateral.
- A rectangle is a quadrilateral.
- A square is a rectangle.

Draw a diagram that represents this hierarchy of quadrilaterals. Use the names **`Quad`**, **`Trap`**, **`Rect`**, and **`Squa`** to represent the shape names. Also, show how the shapes inherit from the **`Object`** class.



2b) (22 points: 22 correctness & style. Use documentation for partial credit.)

You need to develop software that computes the perimeter for each of the three quadrilaterals shown below:



Quadrilaterals with Dimensions
(not drawn to scale)

For full credit, your program should:
- Use all given variable, method, and class names.
- Use an object oriented approach.
- Use encapsulation.
- Use inheritance for the classes **Quad**, **Rect**, and **Squa**.

Fill in missing code inside the boxes and blank lines.

```
public class problem2 {

   public static void main(String args[]) {

      // Instantiate objects from Quad, Rect, and Squa classes:
      //    Constructor for Quad must take all four dimensions as parameters.
      //    Constructor for Rect must take width and height as parameters.
      //    Constructor for Squa must take the length of one side as a
      //    parameter.

      Quad A = new Quad(1,2,3,4);
      Rect B = new Rect(1,4);
      Squa C = new Squa(1);

      // Output the shapes' perimeters:
      A.print();
      B.print();
      C.print();

   } // method main

// class problem2
```

```
// Quadrilaterals
class Quad {

   // Declare instance variables that store the 4 dimensions of the
   // quadrilateral. Call the dimensions s1, s2, s3, and s4.
   private double s1;
   private double s2;
   private double s3;
   private double s4;

   // Use the constructor to assign the variables s1, s2, s3, and s4.
   public Quad(double s1, double s2, double s3, double s4) {
      this.s1 = s1;
      this.s2 = s2;
      this.s3 = s3;
      this.s4 = s4;
   } // constructor Quad

   // Utility Method called perim: computes perimeter.
```

```
   private double perim() {
      return s1+s2+s3+s4;
   } // method perim
```

```
   // Service method called print:
   //    obtains perimeter from perim and prints value.
```

```
   public void print() {
      System.out.println("Perim: "+perim());
   } // method print
```

```
} // class Quad
```

```
// Rectangles
// Class Rect for rectangles: must inherit from superclass Quad
```

```
class Rect extends Quad {
   public Rect(double base, double height) {
      super(base, height, base, height);
   } // constructor Rect
} // class Rect
```

```
// Squares
// Class Squa for squares: must inherit from superclass Rect
```

```
class Squa extends Rect {
   public Squa(double side) {
      super(side,side);
   } // constructor Squa
} // class Squa
```

Problem 3 *35 points: Programming (35 points correctness & style. 2 points for documentation). Both parts (a) and (b) are mandatory.*

Please read this page before proceeding!

The following questions (a) and (b) help you program code that has the following structure:

```
public class problem3 {

   public static void main(String args[]) {

      // body of main

   } // method main

   public static int[] char_counts(String S) {

      // body of char_counts

   } // method char_counts

} // class problem3
```

You need to complete the code for the bodies of methods **char_counts** and **main**. If you get stuck on **char_counts**, you may assume **char_counts** has been correctly programmed so can you finish code for **main**.

Hint: When completed, the code must generate the following output:

```
abc
===
131
110

220
112

310

102
032
400
```

3a) (15 points: 14 for correctness & style, 1 for documentation)

Fill in the code for method **char_counts**, started below. You may assume that method **main** calls **char_counts** from within the body of **main**. The method must count how many times each character **a**, **b**, and **c** appears in **String**. If the method detects an illegal character, the method must *exit* the program with an error message, `Wrong input!`, printed to the user. Method **char_counts** returns a 1-D array of integers. This array stores the number of times **a**, **b**, and **c** appear in string **S** as the first, second, and third elements of the returned array, respectively. Hint: You might wish to use the methods **charAt(index i)** and **length()** from the **StringBuffer** class somewhere in **char_counts**.

```
public static int[] char_counts(String S) {

// count # of times a, b, and c appear in string S

int ac = 0; // count # of a
int bc = 0; // count # of b
int cc = 0; // count # of c

for(int i=0; i<=S.length()-1; i++) {

   if(S.charAt(i) == 'a')
         ++ac;
      else if(S.charAt(i) == 'b')
         ++bc;
      else if(S.charAt(i) == 'c')
         ++cc;
      else {
         System.err.println("Wrong input!");
         System.exit(0);
      }

}

   int counts[] = {ac,bc,cc};
   return counts;

} // method char_counts
```

3b) (20 points: 19 for correctness & style, 1 for documentation)

Fill in the code for method **main**, started below. The initializer list, referenced by **A**, stores arrays of strings. The following code should call **char_counts** to find the character count inside each string element of the initializer list. You must use a 3-D array of integers called **key** to store the returned 1-D array from **char_counts**. Instantiate sizes for **key** only as large as necessary in each dimension. Method **main** must also output each array returned by **char_counts**, as shown on page 7. You must include blank lines as depicted in the output displayed on page 7.

```
public static void main(String[] args) {
   String A[][] = { {"abcbb", "ba"}, {"abba", "accb"},
                    {"aaba"}, {"cca", "cbcbb", "aaaa"} };
   System.out.println("abc");
   System.out.println("===");
```

```
// key stores counts for each character in each string
int[][][] key = new int[4][3][3];

for(int i=0; i<=A.length-1; i++) {

   for(int j=0; j<=A[i].length-1; j++) {

      // get character counts for current string
      key[i][j] = char_counts(A[i][j]);

      for(int k=0; k <= key[i][j].length-1; k++)

         System.out.print(key[i][j][k]);

      System.out.println();

      }

   // Insert blank line between rows of A
   System.out.println();

}
```

```
} // method main
```

Problem 4 *(25 points) Programming Dr. Schwartz's dissertation again! Both parts (a) and (b) are mandatory.*

Recall that addition between two intervals, $\boldsymbol{a} = [\underline{a}, \bar{a}]$ and $\boldsymbol{b} = [\underline{b}, \bar{b}]$, produces $\boldsymbol{a} + \boldsymbol{b} = [\underline{a} + \underline{b}, \bar{a} + \bar{b}]$. You must create a program that adds *arrays* of intervals. Each element of each array is a reference to an **`Interval`** object. Let two 1-D *arrays* of **`Interval`**s, $\boldsymbol{A}$ and $\boldsymbol{B}$, contain the following references:

- Array $\boldsymbol{A}$ has the elements $\boldsymbol{A}_0 = [0, 1]$ and $\boldsymbol{A}_1 = [1, 2]$.
- Array $\boldsymbol{B}$ has the elements $\boldsymbol{B}_0 = [-1, 1]$ and $\boldsymbol{B}_1 = [3, 4]$.

Your program must compute $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$, where "+" implies interval addition between corresponding elements from each array. Thus, $\boldsymbol{C}_i = \boldsymbol{A}_i + \boldsymbol{B}_i$.

This problem requires you to compute the sum of interval arrays $\boldsymbol{A}$ and $\boldsymbol{B}$, shown above. You need to complete the code in parts (a) and (b). This code has the following structure:

```
class Interval {

   // body of Interval

} // class Interval

public class problem4 {

   public static void main(String args[]) {

      // body of main

   } // method main

} // class problem4
```

Complete the code in parts (a) and (b). Hint: Determine the output the program must generate.

4a) (5 points, all correctness & style. Use documentation for partial credit.)

Finish the code for the following **`Interval`** class. Provide a constructor to assign **`min`** and **`max`** values. Provide a method to perform interval addition. This method must return an **`Interval`** Use as few programming statements as possible. You can neglect encapsulation.

```
class Interval {

   double min; // minimum value
   double max; // maximum value

   // Constructor:
```

```
   Interval(double min,double max) {

      // check for correctly entered intervals
      if(min > max) {
         System.err.println("Interval is backwards!");
         System.exit(0);
         }

      // assign instance variables
      this.min = min;
      this.max = max;
      }
```

```
   // Addition method:
```

```
   public Interval addint(Interval B) {

      // Add Interval A to Interval B
      return new Interval(min+B.min, max+B.max);

   } // method addint
```

```
} // class Interval
```

4b) (20 points, all correctness & style. Use documentation for partial credit.)

Finish the code for the following **main** method. You can assume part (a) works correctly, even if you did not finish it. Use 1-D arrays of **Interval**s, referenced by **A** and **B**. Method **main** must compute the sum of the arrays referenced by **A** and **B** and store the result in an another 1-D array, referenced by **C**. Your code must also output each sum of elements in interval notation, e.g., `[0.0,2.0]`.

```
public class problem4 {

   public static void main(String args[]) {

      Interval A[] = new Interval[2];
      A[0] = new Interval(0,1);
      A[1] = new Interval(1,2);

      Interval B[] = new Interval[2];
      B[0] = new Interval(-1,1);
      B[1] = new Interval(3,4);

      Interval C[] = new Interval[2];

      for(int i=0;i<=1;i++) {
         C[i]=A[i].addint(B[i]);
         System.out.println("["+C[i].min+","+C[i].max+"]");
      }

   }

} // class Interval
```

Problem 5 *2 points: Bonus problems (optional). Before working on these, be sure to proofread the entire test! (If you answer 8-9 questions correctly: 2 points; 5–8 correctly: 1 point; 0–4 correctly: 0 points. Regrade requests very unlikely to be accepted.)*

5a) What does "OOP" stand for (in the context of Java programming)?

object oriented programming

5b) Generally, should your code attempt to catch an *error*? (yes or no)

no!

5c) Which of the following identifiers is *not* a reserved word in Java?

`package`, `implement`, `extends`, `interface`

`implements`

5d) What is the Hamming Distance between the DNA sequences AAA and CCC?

3

5e) Name the four nucleotides that make up DNA. You must give the full names.

adenine, cytosine, guanine, thymine

5f) What is a genome?

all the DNA in an organism

5g) What does an amplifier do?

produces an output signal from an input signal
changes signal by lowering or raising

5h) What does a sheet pile do?

shores up soil

5i) Who conceived and wrote the majority of Homework 4 for CS100B, Fall 1999?

Debra Goldberg
Debra
Goldberg