This pledged exam is open text book and closed notes. Different questions have different points associated with them. Because your goal is to maximize your number of points, we recommend that you do not dwell too long on any particular question during your first pass through the exam.

Page 1_____ / 3

Page 2_____ / 18

Page 3_____ / 8

Page 4_____ / 21

Page 5_____ / 15

Page 6_____ / 20

Page 7_____ / 15

Total_____ / 100

Pledge:

1. (3 points) What section of CS101 are you in?

_____ 2 CS101E

_____ 3 0800-0915 Thursday

_____ 4 0930-1045 Thursday

_____ 5 1100-1215 Thursday

_____ 6 1230-1345 Thursday

_____ 7 1400-1515 Thursday

_____ 8 1530-1645 Thursday

_____ 9 1700-1815 Thursday

_____ 10 1830-1945 Thursday

_____ 11 2000-2115 Thursday

2. (3 points) Define and initialize a `Random` variable named `variate`. Variable `variate` should reference a new default constructed `Random` object

3. (3 points) Suppose `variate` is a `Random` variable. Define and initialize an `int` variable `n` so that its initial value is a random number from the inclusive interval 0 … 5.

4. (6 points) Consider the following code segment.

```
Point p = new Point(0, 0);
Point q = new Point(0, 0);
Point r = p;
Point s = (Point) p.clone();
boolean b1 = ( p == q );
boolean b2 = ( p == r );
boolean b3 = ( p == s );
boolean b4 = ( p.equals(q) );
boolean b5 = ( p.equals(r) );
boolean b5 = ( p.equals(s) );
boolean b6 = ( p.equals( "(0, 0)" ) );
System.out.println ("b1 " + b1 + "b2 " + b2 + "b3 " + b3);
System.out.println ("b4 " + b4 + "b5 " + b5 + "b6 " + b6);
```

What does it output?

```
b1 _______________ b2 _______________ b3 _______________

b4 _______________ b5 _______________ b6 _______________
```

5. (6 points) Suppose `j` and `k` are previously defined and initialized integer variables. Write an `if` statement that sets variables `j` and `k` to 1 when `j` is greater than `k`; otherwise it sets `j` and `k` to 2.

6. (4 points) Write a code segment that displays the 100 possible combinations of adding two numbers from the inclusive interval 1 … 10. For example, when 3 and 5 are respectively considered as the left and right operands, the display should be

```
3 + 5 = 8
```

7. (4 points) Suppose `counter` is an *already defined* `int` variable. Also suppose `Scanner` variable `stdin` has *already been defined and initialized* to represent standard input, where the input values are all numbers. Write a code segment that reads all the values from standard input and assigns to `counter` the number of values that were read.

8. (3 points) Define a new `int[]` variable `list` that references an array composed of the values 2, 4, and 6.

9. (3 points) Define a new `int[]` variable `list` that references a *new* array with five elements.

10. (3 points) Define a new `int[]` variable `list` that references a *new* array with zero elements.

11. (3 points) Suppose `list` is an initialized, non-`null`, `int[]` variable referencing an array with at least one element. Write a statement that assigns 10 to the *first* element of `list`.

12. (3 points) Suppose `list` is an initialized, non-`null`, `int[]` variable referencing an array with at least one element. Write a statement that assigns 10 to the *last* element of `list`.

13. (6 points) Consider the following legal program.

```
public class Think {
   public void static mystery(int n) {
      n = (int) ( Math.pow(n, 2.02) * Math.sin( Math.sqrt(n) ) );
   }

   public static void main(String[] args) {
      int n = 5;
      mystery(n);
      System.out.println("n = " + n);
   }
}
```

What does it output?

```
n = _____
```

Why?

Questions 14–23 deal with a class named `Calculator`. Class `Calculator` has the following attribute.

- `private double runningTotal` – represents the value (running total) of the current calculation.

Class `Calculator` is to have the following `public` methods.

- `Calculator()` – default constructor that configures the calculator to have a running total of 0.
- `Calculator(double v)` – specific constructor that configures the calculator to have a running total of `v`.
- `void clear()` – sets the running total of the calculator to 0.
- `double getTotal()` – returns the running total of the calculator.
- `void add(double v)` – increments the running total by amount `v`.
- `void divide(double v)` – if v is non-zero, then its sets the running total of the calculator to the value of the current running total divided by `v`. Otherwise, it sets the running total of the calculator to the `double` value `Double.NaN`. If you are interested, `Double.NaN` is discussed on page 66 of the text.
- `Object clone()` – returns a new `Calculator` with the same running total as this calculator.
- `boolean equals(Object v)` – returns whether its parameter is a `Calculator` with the same running total as this calculator.
- `String toString()` – returns a `String` representation of the running total. One of the `String` class `valueOf()` methods may prove helpful here. They are discussed on page 878 of the text.

14. (5 points) Implement the `Calculator` default constructor using an assignment statement to configure the running total attribute.

15. (5 points) Implement method `clear()`.

16. (5 points) Implement the `Calculator` default constructor using method `clear()` to configure the running total.

17. (5 points) Implement `Calculator` method `getTotal()`.

18. (5 points) Implement `Calculator` method `add(double v)`.

19. (5 points) Implement `Calculator` method `divide(double v)`.

20. (5 points) Implement `Calculator` method `clone()`.

21. (5 points) Implement `Calculator` method `equals(Object v)`.

22. (5 points) Implement `Calculator` method `toString()`.

23. (5 points) Suppose `list` is a `Calculator[]` array variable. Write a code segment that modifies the calculator associated with `list[i]`. The modification should add 5 to its running total.