This exam is closed-book, closed-notes, no electronic devices allowed. The exception is the “sage page” on which you may have notes to consult during the exam. Answer questions on the pages of the exam. Do not unstaple the pages of this exam, nor should you attach any other pages to the exam. You are welcome to use the blank space of the exam for any scratch work.

Your work must be legible. Work that is difficult to read will receive no credit. Do not dwell over punctuation or exact syntax in code; however, be sure to indent your code to show its structure.

You must sign the pledge below for your exam to count. Any cheating will cause the students involved to receive an F for this course. Other action may be taken. If you need to leave the room for any reason prior to turning in your exam, you must give your exam and any electronic devices with a proctor.

You must fill in your identifying information correctly. Failure to do so is grounds for a zero on this exam. When you reach this point in the instructions, please give the instructor or one of the proctors a meaningful glance.

<table>
<thead>
<tr>
<th>Print clearly the following information:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (print clearly):</td>
</tr>
<tr>
<td>Student 6-digit ID (one digit per box):</td>
</tr>
</tbody>
</table>

Your answers below tell us where to return your graded exam.

What time do you arrive in studio/lab? (circle one) 11:30 1:00 2:30 4:00

Which Urbauer lab? (your best guess, circle one) 214 216 218 222

Pledge: On my honor, I have neither given nor received any unauthorized aid on this exam.

Signed: ____________________________________________

(Be sure you filled in your information in the box above!)
<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Possible Points</th>
<th>Received Points</th>
<th>Grader Initials</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. (15 points) Recalling the Module 5 exercises, the following problem should be familiar. You need not worry about handling null. You may write any of the methods recursively, or not, at your preference. You will write three methods: concat2, concatN, and join such that:

```java
public static void main(String[] args) {
    System.out.println(concat2("echo").
        for (int i = 0; i < 10; i++) {
        System.out.println(i + " " + concatN("X", i));
    }
    System.out.println(join("I love computer science.".split(" "), "<<>>").
```

will output

echo echo
echo
0
1 X
2 XX
3 "XXX"
4 "XXXX"
5 "XXXXX"
6 "XXXXXX"
7 "XXXXXXX"
8 "XXXXXXXX"
9 "XXXXXXXXX"
I<<>>love<<>>computer<<>>science.

(a) (5 points) Complete the code below to produce a String that is two copies of s concatenated together.

```java
public static String concat2(String s) {

    }
```

Continued on next page...
(b) (5 points) Complete the code below to produce a String that is \( n \) copies of \( s \) concatenated together.

```java
public static String concatN(String s, int n) {
}
```

(c) (5 points) Complete the code below to join an array of Strings much like an inverse of the `split(...)` method on String. Between each item in the array, you should place the value of the `joiner` parameter.

```java
public static String join(String[] array, String joiner) {
}
```
2. (20 points) Throughout the semester we have asked you to format your doubles to some number of digits after the decimal point. To provide the functionality of producing a double to a specified $n$ digits after the decimal point, we will have you write two methods: `intPower` and `toNDigitsAfterDecimal` such that:

```java
for (int i = 0; i <= 8; ++i) {
    System.out.println(i + " " + toNDigitsAfterDecimal(Math.PI, i));
}
```

will output

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\text{toNDigitsAfterDecimal}(\mathbf{\pi}, i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.0</td>
</tr>
<tr>
<td>1</td>
<td>3.1</td>
</tr>
<tr>
<td>2</td>
<td>3.14</td>
</tr>
<tr>
<td>3</td>
<td>3.141</td>
</tr>
<tr>
<td>4</td>
<td>3.1415</td>
</tr>
<tr>
<td>5</td>
<td>3.14159</td>
</tr>
<tr>
<td>6</td>
<td>3.141592</td>
</tr>
<tr>
<td>7</td>
<td>3.1415926</td>
</tr>
<tr>
<td>8</td>
<td>3.14159265</td>
</tr>
</tbody>
</table>

(a) (10 points) Complete the code below WITHOUT using the `Math.pow(double x, double y)` function which produces $b^{a}$. You should write clean code that would handle any $exponent \geq 0$ ignoring integer overflow. You may write this method recursively or not at your preference.

```java
public static int intPower(int base, int exponent) {
    // Continued on next page...
}
```

---

1. Do not be confused by 0 digits after the decimal point outputting 3.0. This is just how Java prints double values equal to values like three.

2. You need not worry about negative numbers or integer overflow. We won’t grade on that behavior.
(b) (10 points) Complete the code so that it produces the result of truncating \( d \) to \( n \) digits after the decimal point.\(^3\)

```java
public static double toNDigitsAfterDecimal(double d, int n) {
}
```

\(^3\)You need not worry about negative numbers or integer overflow. For \( n < 0 \) and \( n > 8 \), you can do whatever you like. We won’t grade on that behavior.
The Collatz Conjecture states that if you start with an integer greater than or equal to 1, and repeatedly apply the function described in the comic, that you will eventually reach 1. To test this conjecture you will complete two methods: isOdd and collatz.

(a) (5 points) Write code to calculate if the parameter n is odd so that it works properly for any value of n.

```java
public static boolean isOdd(int n) {
    // Code goes here
}
```

Credit: xkcd, available by fair-use policy

4Your friends might or might not stop calling you.
(b) (5 points)

\[
collatz(n) = \begin{cases} 
1, & \text{if } n = 1 \\
collatz(n/2), & \text{if } n \text{ is even} \\
collatz(3n + 1), & \text{if } n \text{ is odd}
\end{cases}
\]

First, show the complete evaluation of \( \text{collatz}(5) \), using substitution as in studio, beginning with:

\[
collatz(5) =
\]

(c) (5 points) Now complete the code below to implement the above recurrence:

\[
\text{public static int collatz(int n) }
\]

\[
\}
\]

\footnote{For }
4. (15 points) Consider the following formula for the energy of a mass \( m \) at rest with speed of light denoted as \( c \):

\[
E_0 = mc^2
\]

(a) (5 points) Complete the code below to compute the square of the parameter value by multiplying it by itself:

```java
private static double square(double value) {
    // Code here
}
```

(b) (5 points) Below, write the code to compute the rest energy \( E_0 \):

```java
// speed of light in a vacuum in meters per second, commonly denoted c
public static final double SPEED_OF_LIGHT = 299_792_458;

public static double restEnergy(double mass) {
    // Code here
}
```

(c) (5 points) Consider the formula for the energy of a mass \( m \) with velocity \( v \):

\[
E = \frac{mc^2}{\sqrt{1 - \frac{v^2}{c^2}}}
\]

Recall the API:

```java
Math.sqrt(a)
```

produces the square root of \( a \). Below, write the code to compute the energy \( E \):

```java
public static double energy(double mass, double velocity) {
    // Code here
}
```
5. (20 points) Recall the API for drawing a circle is as follows:

\[
\text{StdDraw.circle}(x, y, r)
\]
draws a circle centered at coordinate \((x,y)\) whose radius is \(r\). Consider the following image:

In this problem you develop a recursive solution to drawing the image shown above.

(a) (5 points) Describe the substructure of the image you see above.

(b) (5 points) Describe the base case of the recursion present in the image above.

Continued on next page...
(c) (10 points) Complete the recursiveCircles method below so that it draws something resembling the image above. Note: You need not worry about the smaller circles being exactly size in the image above. Choose a reasonable amount to shrink them by and you should be fine.

```java
public static void main(String[] args) {
    StdDraw.setXscale(-1, +1);
    StdDraw.setYscale(-1, +1);
    recursiveCircles(0, 0, 0.5);
}

public static void recursiveCircles(double x, double y, double radius) {
}
```
6. (15 points) The Wallis Product states that:
\[
\frac{\pi}{2} \approx \prod_{k=1}^{n} \left( \frac{2k \cdot 2k}{2k - 1 \cdot 2k + 1} \right) = 2 \cdot \frac{2}{1} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9} \cdots \frac{2n}{2n - 1} \cdot \frac{2n}{2n + 1}
\]

(a) (5 points) Draw a box around the recursive substructure of the series above.

(b) (10 points) Write recursive code for `computePiOver2` below corresponding to the definition above.

```java
private static double computePiOver2(int n) {

```

```java
}
```