This exam is closed-book, closed-notes, no electronic devices allowed. The exception is the “cheat sheet” 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>Problem Number</th>
<th>Possible Points</th>
<th>Received Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>

**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!)
1. **(20 points)** Consider the drawing shown below:

- The canvas is a Sedgewick-standard 1.0 × 1.0 canvas, with (0, 0) as its lower-left-hand corner and (1, 1) as its upper-right-hand corner.
- The largest circle you see has radius 0.5.
- The radius of each circle you see is 90% of (0.9 ×) the radius of its enclosing circle.
- Only those circles whose radii exceed 0.01 are drawn.
- Recall that the syntax for drawing a circle centered at (x, y) with radius r is:
  \[ \text{StdDraw.circle}(x, y, r); \]

Based on the above, answer the following questions:

(a) **(5 points)** Below, explain the recursive substructure of the picture:

*Continued on next page...*
(b) (10 points) Below, complete the recursive circles method and the (nonrecursive) main method so that they draw the picture exactly as described above. Do not worry about the pen radius or the color used to draw the circles.

public class Circles {  // (8 points)

    // Fill in the blank portions below as needed for your solution
    //
    public static ________ circles(__________________________) {
        //
        // This method must be recursive. Use NO iteration
        //
    }

    public static void main(String[] args) {  // (2 points)

    }

    }

(c) (5 points) In your recursive method above, draw a box around the base case.
2. (30 points) Consider the following function:

\[ f(n) = \begin{cases} 
  n + (n - 3) + (n - 6) + \ldots + 0, & \text{if } n \geq 0 \\
  0, & \text{otherwise} 
\end{cases} \]

To illustrate this computation, the following table shows some sample values:

<table>
<thead>
<tr>
<th>n</th>
<th>f(n)</th>
<th>n</th>
<th>f(n)</th>
<th>n</th>
<th>f(n)</th>
<th>n</th>
<th>f(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>22</td>
<td>9</td>
<td>18</td>
<td>8</td>
<td>15</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-131</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) (10 points) Below, complete the nonrecursive function that computes \( f(n) \). Do not write a main or surround the code with a class declaration. You need only complete the function \( f \).

```java
//
// Fill in the blank portions below as needed for your solution
//
public static ________ f(______________________) {
    //
    // This method must NOT be recursive. Use iteration
    //
```
(b) (10 points) Now write a recursive solution for \( f \):

```java
//
// Fill in the blank portions below as needed for your solution
//
public static _______ f(__________________________) {
    //
    // This method must be recursive. Use NO iteration
    //
```

(c) (10 points) Below, show the evaluation of \( f(8) \) using the substitution method:
3. (25 points) Consider the following functions:\(^1\)

```java
class Example {
    public static int g(int a, int b, int c) {
        return a*b + c;
    }
    public static int h(int a, int b, int c) {
        return a*b + c;
    }
}
```

Now consider the following expression:

g(1, h(2, 3, 4), 5)

(a) (2 points) Below, circle the function that executes first:

```
g h
```

(b) (4 points) Assume that the runtime stack is empty before the above expression executes. Circle the value that is the first to be pushed on the stack:

```
1 2 3 4 5 10 15
```

(c) (4 points) Circle the value that is the last to be pushed onto the stack:

```
1 2 3 4 5 10 15
```

(d) (15 points) Place values on the following stack diagrams to correspond to the situations described below:

```
(i) (ii) (iii)
```

i. (5 points) Just prior to h beginning its execution

ii. (5 points) Just after h returns from its execution

iii. (5 points) Just prior to g beginning its execution

---

\(^1\)These two functions are identical, but they have different names to make this question easier to pose and answer.
4. (25 points) Consider the following user story

When a group of people (hereafter, called a party) approaches a restaurant, there is sometimes a wait for a table to become available to accommodate the party. In response to a party arriving and not being immediately seated, the restaurant instantiates a Reservation object for the party. In terms of design we say Reservation has-a:

name of the party, which is the last name of the person who arrives and asks the restaurant for a Reservation.

arrival number of the party, which is an integer assigned by the restaurant to the party when the Reservation is created. This is not the number of people in the party—we won’t be keeping track of that.

seating preference expressed by the party for whether they wish to be seated outside or inside.

A Reservation requires the above information to be instantiated. Other aspects of this problem are as follows:

- Arrival numbers are generated sequentially by the restaurant, so that if two parties arrive such that \( a \) obtained its reservation prior to \( b \), then we can be sure that \( a < b \).
- Although multiple parties may share the same name, each party’s arrival number is unique.
- When a table is ready, the reservation is called by number, not by name.
- Parties are seated methodically according to their reservation number. For all \( a < b \), the party with arrival number \( a \) will be seated before the party with arrival number \( b \).
- All diners in a given party may not arrive at the same time. Late-arriving diners may supply their own name, but they are issued a (new) Reservation with the same arrival number and seating preference as the first Reservation instantiated for the party. You should provide a constructor to make this easy and robust.
- We do not keep track of the number of people in any party. All parties can be seated at any available table in this restaurant.

In terms of behavior, our Reservation object should offer the following functionality:

- It should be possible to find out the name and arrival number of a Reservation.
- It should be possible to find out if a Reservation wants to be seated inside.
- For a given Reservation, it should be possible to find out whether this Reservation will be seated before another Reservation.
- For a given Reservation, it should be possible to determine whether this Reservation has the same seating preference (be it outside, or inside) as another Reservation.

Continued on next page...
(a) (3 points) Below, list the instance variables and their types. Choose a type for each instance variable that best fits its purpose within your class.

(b) (2 points) Do not write equals or hashCode methods for your code. However, below please explain upon which instance variable(s) equality should be based, and provide justification for your answer:

(c) (20 points) Below, and on the facing page, write the code for your Reservation class, placing elements of your class between the comments as indicated.

```java
public class Reservation {

    //
    // (1 point) Instance variable(s) below here.
    // Use final where appropriate.
    //

    //
    // (5 points) Constructor(s) below here
    //
```
//
// (4 points) toString() below here
//
//
// (10 points) Other methods below here. DO NOT provide equals or hashCode
//

// use back of this page if necessary
}