This exam is closed-book, closed-notes, no electronic devices allowed except for downloading the exam and sending the completed exam. The exception is the “cheat sheet” on which you may have notes to consult during the exam.

The professor will be available online to answer any questions you have during the afternoon from 2-4 PM central time. If for any reason you cannot reach him, just write down your assumptions and proceed as best you can.

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.

You must fill in your identifying information correctly.

To submit your exam, mail the scanned pages to roncytron@gmail.com.

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Possible Points</th>
<th>Received Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</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) In Module 8, you used a LinkedList implementation to construct a Polynomial abstract data type. In this course we have also seen the Stack ADT. To simplify matters, let’s say that Stack offers only the following two methods:

push(x) places x on top of the stack
pop() removes and returns the topmost element on the stack

The type of x and the type of the result returned by pop are unspecified above. For this problem, you may choose to use a parametric type for the contents of your stack, or you may assume that the ADT uses only the String data type.

(a) (1 points) Circle one of the following:
   - I am using a parametric type for my stack’s elements
   - I am designing a stack that can hold only Strings.

(b) (9 points) Below, define a Java interface that expresses the Stack abstract data type:

(c) (15 points) On the next page, write a Java class ListBasedStack that implements the Stack interface using a single List object as its only instance variable. You must provide the following methods:
   - (1 points) the constructor takes no parameters.
   - (4 points) a method push that effectively places its parameter’s value on the top of the stack
   - (4 points) a method pop that effectively returns the top element on the stack.
   - (5 points) a toString() method that returns a description of the contents of the stack (the format is up to you)

Continued on next page...
2. (40 points) A ListItem is defined below:

```java
public class ListItem {
    public int number;
    public ListItem next;
    public ListItem prev;

    public ListItem(int n) {
        this.number = n;
        this.next = null;
        this.prev = null;
    }
}
```

(a) (10 points) A code sequence is shown below. In the space to the right of the code sequence, draw the instantiated objects and illustrate how references point to objects using arrows. Use circles to depict ListItem objects and rectangles to hold the source of a ListItem reference. Be sure to show all references (including a–e).

```java
ListItem a = new ListItem(1);
ListItem b = new ListItem(2);
ListItem c = new ListItem(3);
ListItem d = new ListItem(4);
ListItem e = new ListItem(5);

a.next = c;
c.next = e;
a.prev = b;
e.prev = d;
b.prev = a.next.next.prev;
```

(b) (20 points) A ListItem object as defined above is stored in memory as three contiguous memory cells. The first cell corresponds to the number, the second cell contains the memory address of the next ListItem, and the third contains the memory address of the prev (previous) ListItem.

Using the table below, and starting with memory location 109, write below the contents of the list according to each number stored in its ListItem, following next references until you reach null. For example, if the number values you find in memory as you go through the list are 3, 21, and 4, in that order, then your answer for this part is simply 3, 21, 4

Continued on next page...
List starting at 109:

(c) (10 points) Draw an objects and references diagram (circles, rectangles, and arrows) for the structure you find starting at address 106. Be sure to show all references.
3. (20 points) For each of the scenarios below, place an X in the column whose associated abstract data type (ADT) is best suited for the scenario.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>List</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>The results of 10 coin tosses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The kinds of coins you have in your pockets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The team roster of the St Louis Blues, in order of their joining the team</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The books you had to buy for classes this semester</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The coefficients of a polynomial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The names of variables used in a polynomial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The signatures of the methods offered by a particular Java class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The menu of foods available at the DUC (campus eatery)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The steps needed to bake a cake</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The ingredients needed to bake a cake</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. (15 points)

(a) (5 points) When written in Java, how can you tell the difference between a Java class and an interface?

(b) (5 points) In terms of intent and design, why are interfaces useful? What particular needs do they address?

(c) (5 points) Let’s say that two interfaces a and b are distinct if they do not offer exactly the same methods.

If a class offers 5 methods, how many distinct interfaces could be defined such that the class, as-is, could implement any of those interfaces? Explain your reasoning.