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.

During the live session, Ze’ev will be online to answer any questions you have. 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.

---

**Print clearly** the following information:

<table>
<thead>
<tr>
<th>Name (print clearly):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 6-digit ID (print <em>really</em> clearly):</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Possible Points</th>
<th>Received Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30</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. (30 points) In this problem we are interested in computing the number of times the integer 131 occurs in an array of int s over some specified range. As is typical in computer science, if we specify the range as running from $i$ to $j$, then the range using its element $i$ and all other elements up to, but not including, element $j$. Consider the array:

```java
int[] a = new int[] { 131, 5, 7, 132, 14, 131}
```

In this example $a.length=6$. Some examples of our computation are as follows:

- $\text{num131s}(a, 0, a.length)$ should return 2. This range begins at index 0 (the array’s first element) and goes up to but does not include index 6. Thus, this range considers all of the array’s elements. Because there are two occurrences of 131 in that range, we return 2.
- $\text{num131s}(a, 2, a.length)$ should return 1. The range here is the same as previously, except the first two elements (at indexes 0 and 1) are skipped. Those four elements of the array contain 131 only once, so we return 1.
- $\text{num131s}(a, 1, 2)$ should return 0. This range considers only $a[1]$. There is no 131 in that range.

(a) (5 points) Below, fill in the missing blanks as indicated:

```java
//
// Fill in what is missing below for the method declaration
//
public static _______ num131s(_____________________________) {
    // No code here, just fill in the above blanks
    // Even though this may not be the right return value,
    // complete the blank below with a return value that matches
    // your choice of return type above
    return _______________
}
```

(b) (5 points) Let’s try to find some recursive substructure in this num131s problem. Consider a sample array $a$ whose contents are shown below, where the spacing between the integers is just to provide some room:

```
131 5 7 132 14 131
```

A call to $\text{num131s}(a)$ should return 2, since there are two 131s in the above array. A copy of int[] a is shown below. If we view the entire array as the larger problem, draw a box around the next smallest instance of that larger problem:

```
131 5 7 132 14 131
```

Continued on next page...
(c) (5 points) What answer should be returned for the smaller problem you have identified? The smaller problem is the same int[] array but using the range associated with the box you drew above.

(d) (10 points) Based on the recursive substructure you have identified, describe in words a recursive approach to computing num131s(a, start, end) on some int[] a. You do not need to write code for this: a description in English suffices.

(e) (5 points) What is the base case of your recursive computation, and what value should it return?
2. (20 points) Using recursion or iteration (your choice), write below an implementation of `num131s`:
3. (20 points) Consider the following function:

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

Now consider the following expression:

```
h(2, 3, 4)
```

(a) (5 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:

```
2   3   4
```

(b) (5 points) Circle the value that is the last to be pushed onto the stack:

```
2   3   4
```

(c) (10 points) Place values on the following stack diagrams to correspond to the situations described below:

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

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

ii. (5 points) Just after h returns from its execution
4. (30 points) Consider the following user story:

A Building has-a

- number of floors
- a floor-height (in integral inches): each floor is of this height.
- a latitude: the north-south position of the center of this building. For example, the latitude value for the Hockey Hall of Fame in Toronto, Ontario, Canada is ??????
- a longitude: its east-west position of the building’s center. For the Hockey Hall of Fame, this value would be ??????????

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

- There should be getters, but no setters, for each of the Building’s instance variables.
- It should be possible to find the total height of the building in inches.
- Given an object of some height in inches, it should be possible to tell whether that object could stand on the floor of this Building.
- Given some other Building b, it should be possible to tell whether this Building is taller than b.

Note that the above requirements do not print out anything. Your methods therefore should not print anything, but should instead return appropriately typed values to provide the functionality.

(a) (6 points) Although you will not generate hashCode or equals, on which of the above “has-a”s would you base those methods, and why?

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

```java
public class Building {

    //
    // (10 point) Instance variable(s) below here. For each instance variable, chose a type that best fits its purpose and usage.
```
//
// (5 points) Constructor below here.
//

//
// (4 points) toString() below here.
// (something very simple: no need to include all instance variables)

//
// (10 points) Other methods below here. DO NOT provide equals or hashCode.
//