Write your name and Cornell netid. There are 5 questions plus one extra credit question on 7 numbered pages. Check now that you have all the pages. Write your answers in the boxes provided. Use the back of the pages for workspace. Ambiguous answers will be considered incorrect. The exam is closed book and closed notes. Do not begin until instructed. You have 90 minutes. Good luck!

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>ext</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>/20</td>
<td>/20</td>
<td>/20</td>
<td>/20</td>
<td>/20</td>
<td>/5</td>
<td></td>
</tr>
<tr>
<td>Grader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. (20 points) A florist shop uses class Bouquet to manage a static ArrayList<Bouquet> in which all the bouquets currently on sale are listed. Here’s the class (note: it uses an “enumerated” list of colors. This just means that the listed names (red, yellow...) can be used as “values” of objects of type Color).

```java
class Bouquet {
    enum Color { red, yellow, orange, purple }; // The list of colors we use
    public Color primaryColor; // primaryColor of this bouquet
    public int price; // price in dollars
    public String name; // Bouquet name in the catalog
    public long inventoryId; // An inventory id number

    private static ArrayList<Bouquet> theInventory; // Bouquets in stock
}
```

(a) [5 points] Bouquet has several instance fields (primaryColor, price, etc) but theInventory, which lists all the Bouquets in the store, is static. Give one good reason that we might prefer that a store-wide inventory not be an instance field.

(b) [10 points] Write the method to implement its specification. Make your code match the comments.

```java
/** Return the subset of the inventory for which c is the primary color */
public static ArrayList<Bouquet> floralOptions(Color c) {
    // Allocate a new ArrayList<Bouquet> called matching.

    // Iterate over the inventory, adding to “matching” the bouquets that match c

    // Return the new list.
}
```
(c) [5 points] If you modify this code to say ArrayList instead of ArrayList<Bouquet>, everything seems to work identically. Explain why cs2110 students use ArrayList<Bouquet> even though ArrayList also works.
2. (20 points) True or false?

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>b</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>c</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>d</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>e</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>f</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>g</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>h</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>i</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>j</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>k</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>l</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>m</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>n</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>o</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>p</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>q</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>r</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>s</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>t</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
3. (20 points) Suppose we create a min-heap (of maximum size seven) by inserting five elements 71, 3, 8, 15, -7 in the order shown (leaving room for two more). Draw the resulting heap first as a tree (exactly as seen in class) and as a vector (again, using the vector representation of a heap seen in class).

(a) 5 points. The heap:

(b) 5 points. The corresponding vector (put a slash through “empty” elements):

(c) 5 points. Now show the vector after we call poll() once, to extract the smallest element.
(d) 5 points. Suppose that you are looking at element $k$ in the vector representation of a heap. Write a public static method that computes the index of the element containing the *parent* of $k$. (By index we mean that $k$ is an offset into the vector representing the heap). The root has no parent; return 0 in this case. *Note: this has a simple answer. Very complex but correct answers won’t get a lot of partial credit.*

```java
/** Given a heap in vector representation, computes the index of the element containing the parent of $k$. Returns 0 if called for the root. */
public static int parent(int k) {

}
```
4. (20 points) Suppose that we are given a binary tree in which nodes have public instance fields `left` and `right` pointing to the left and right children (or `null`, if there is no child on that side), and with user-defined `equals()`, `compareTo()`, and `hashCode()` methods. Write method `treeEquality` to implement the specification given below. You can assume that the input is a valid tree.

a. [15 points] *Hint: this can be done very easily! We’ll deduct for inelegant, complex code.*

```java
/** Return true iff ta and tb are identical trees —they are both null or both 
 trees with the same shape and corresponding nodes equal. */
public static boolean treeEquality(TreeNode ta, TreeNode tb) {
    // Your implementation here
}
```
b. [5 points] We overload treeEquality with this method:

```java
/**
 * sets is a set of M balanced STs, each containing N nodes (for some M ≥ 0, N ≥ 0).
 * Return true iff for all x, y in sets, treeEquality(x, y) is tree */

public static Boolean treeEquality(Set<BST> sets) { ... }
```

In English, describe an efficient way to use the overload of treeEquality() from part a to solve this problem (without changes to the method from part (a), and without giving the code for this new version of treeEquality). Keeping your answer short, now tell us what will be the worst-case O() time complexity of the new method, expressed in terms of N and M. Justify your answer.
5. (20 points) You are writing code for a computer game in which the user explores a cave system consisting of caverns linked by tunnels, gathering weapons and treasures while fighting monsters. The game ends when the user is killed, or wins by discovering a special room containing the Grail of Hwynza. The cave is represented as an undirected graph. The Grail is within a finite distance of the entrance and each cavern is connected to finitely many other caverns.

a. [5 points] There is a famous way of escaping from any maze: Keep your hand in contact with the left wall and just keep walking. Can this method be used to hunt for the Grail? If yes, give a small proof that justifies your answer; if no, draw a cave system in which this method would fail.

b. [5 points] Suppose that the cave is of infinite extent (but that any cavern has finitely many adjacent caverns). Would a depth-first search strategy be certain to find the Grail? Again, don’t just say yes or no: give a proof of your answer.
d. [5 points]. Again, suppose that the cave is of infinite extent but that a cavern has finitely many neighbors. Would a breadth-first search strategy be certain to find the Grail? Prove your answer.

e. [5 points]. In a finite cave system with $N$ caverns in which the Grail is hidden in a random cavern, what would be the average $O()$ complexity of finding the Grail, given that “visiting a room” is the operation we wish to count (e.g. “visiting” has cost 1)? Again, justify your answer.
ext. [3+2 points extra credit] A clique of a graph G is a subgraph C such that for every two distinct vertices v₁ and v₂ in C, (v₁, v₂) is an edge of C.

A graph is maintained in adjacency-list form: it is given by variable G of type List<Vertex>, where class Vertex has a public instance field List<Vertex> neighbors, giving the vertices to which it has an edge.

For 3 points of extra credit, correctly complete the body of method cliqueTest, below.

For 2 additional points of extra credit write a second method maxCliqueSize, specified below.

Assume there is an existing method public static Set<T> intersect(Set<T> a, Set<T> b), and a second public static Set<T> union(Set<T> a, Set<T> b); the first returns a new Set<T> that is the intersection of its arguments, and the latter returns the union.

```java
/** Return true iff the vertices in v form a clique of G*/
public static boolean cliqueTest(List<Vertex> G, List<Vertex> v) {
```

}``
/** Return the largest integer k such that G contains a clique of size k.
   * Precondition: G has at least one vertex. */
 public static int maxCliqueSize(List<Vertex> G) {

}