CS2102, B10

Exam 1

Name:

Problem	Points	Score
1	40	
2	30	
3	30	
	Total	

You have 50 minutes to complete the problems on the following pages. There should be sufficient space provided for your answers.

You do not need to show templates or an `Examples` class.

If a problem asks you to create a class hierarchy, we are looking for the interfaces, classes, and abstract classes that you would create for the problem. In particular:

- Include **implements** and **extends** statements
- Include field names and types
- Include method headers (names, return type, and input parameter types)
- Full credit requires outlining all constructs in Java (so that the specific types and implements/extends are clear). Class diagrams alone may earn partial credit.
- You may omit constructors
- You may omit method bodies unless a question asks otherwise
- You may omit the `Examples` class unless a question asks otherwise

---

1. (40 points) You are developing a class hierarchy for student records at a university. The following concrete classes and their fields capture the essential information about three kinds of students:

```
class Undergraduate {
  String name;
  int graduationYear;
  int creditsPassed;
}

class GradStudent {
  String name;
  int enteringYear;
  String dept;
  int creditsPassed;
}

class ContinuingEducationStudent {
  String name;
  String dept;
  int creditsPassed;
}
```

The university needs the following functions on students:

- *addCredits* which consumes a student and a number of credits and produces a student with the credits increased by the given amount.
- *inDept* which consumes a graduate or continuing-education student and a department and produces a boolean indicating whether the student is in the given department.
- *canGraduate* which consumes a student and returns a boolean indicating whether the student has enough credits for a degree. The formula is significantly different for each kind of student.

Create a class hierarchy around the given classes that includes the required functions and shares common code where feasible. You may move fields out of the given classes. Edit the given classes by writing on the text above (cross out removed fields, add implements/extends, etc).

Use the rest of this page and the next (blank) page for any new interfaces/classes/abstract classes.

(exam continues next page)

(exam continues next page)

2. (30 points) Draw examples (pictures, not code) satisfying each of the following descriptions:

(a) (15 points) A balanced binary-search tree containing the numbers 1 through 7.

(b) (15 points) A heap containing the numbers 1 through 7 that is NOT balanced.

(exam continues next page)

3. (30 points) Recall the following interface for Priority Queues:

```
interface PriorityQueue {
  // adds the given element to the priority queue
  PriorityQueue addElt(int elt);

  // removes smallest element from priority queue
  PriorityQueue remMinElt();

  // returns, but does not remove, smallest elt in priority queue
  int getMinElt();
}
```

Assume you want to use lists (not necessarily sorted) to implement the PriorityQueue interface. Propose a set of test cases designed to check that your List classes accurately capture a priority queue.

You do **not** need to write your test cases in Java. For each test, provide a few words describing what the test is checking, the test expression, and the expected output. Your tests may refer to variables that you define in prose (i.e., "L1 is a list containing 2, 5, 4 in that order"). We are more interested in *what* you choose to test than your ability to write the tests in Java.

(end of exam)