Grades for the final will be posted on the CMS tomorrow night, late. Grades for the course will take a few days more. You can look at your final when you return in the Spring. Thanks for taking the course.

HAVE A NICE WINTER BREAK!

Question 0. _____	(out of 01)
Question 1. _____	(out of 12)
Question 2. _____	(out of 12)
Question 3. _____	(out of 15)
Question 4. _____	(out of 12)
Question 5. _____	(out of 24)
Question 6. _____	(out of 12)
Question 7. _____	(out of 12)
Total _____	(out of 100)

You have 2.5 hours to complete the questions in this exam, which are numbered 0..7. Please glance through the whole exam before starting. The exam is worth 100 points.

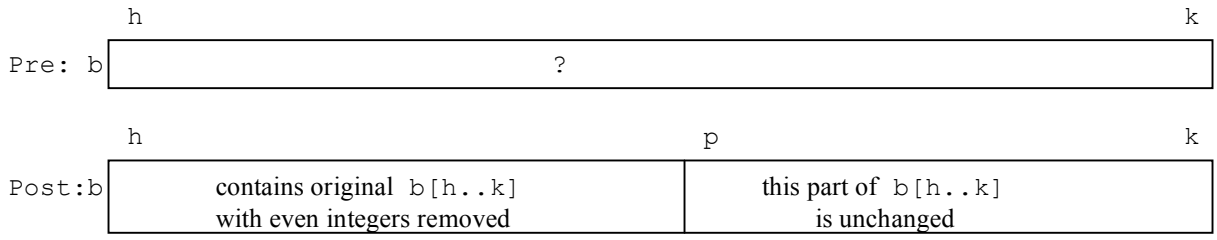
Question 0 (1 point). Print your name and **net id** at the top of each page. Please make them legible.

Question 1 (12 points). Algorithms.
Write algorithm `partition` as a function, complete with method header (giving the parameters, for example) and a suitable specification —a precondition and a postcondition. Remember that the function has to return an **int**.

You may give the pre- and post-conditions as formulas or as pictures. You *must* write an invariant, and the loop you write *must* be developed from the invariant and pre- and post-conditions.

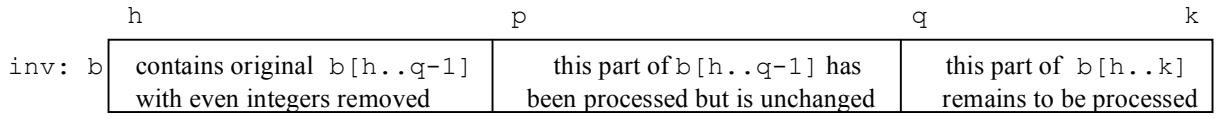
If you have to swap two variables `x` and `y` (say), just write "swap x and y"; you need not write the sequence of three statements to swap them.

Question 2 (12 points). Loops. Write a single loop (with initialization) to remove even integers from **int** array segment b[h..k], as indicated by the following precondition and postcondition.



For example, `for b[h..k] = {1, 7, 6, 8, 3, 2, 5}`, execution of the algorithm sets `p` to `h+4` (because there are 4 odd values) and changes `b[h..k]` to this: `{1, 7, 3, 5, 3, 2, 5}`. The first four values in `b[h..k]` are the elements of the original `b[h..k]` but with the even integers removed. The last part `b[p..k]` is unchanged.

Write one loop (with initialization) that performs the task. Do not declare any variables; assume they are all declared. You **must** use the invariant shown below. Do not use any variables (except those shown in the invariant) outside the loop. Read the invariant carefully before proceeding. Note that we are *not* asking for a method, so do not write return statements. Follow the four loopy questions, and you will do well on this question.



Question 3 (15 points). Arrays. (a) Execute the following assignment, drawing all the objects created and properly assigning to all variables. Put the answer below or on the back of the next page.

```
int[][]  y=  new int[][]{ {3, 2}, {4, 5, 6}, {}};
```

Question (b) Write procedure `swap` below, which swaps two non-overlapping square sections of b. For example, suppose n = 2 and h = 0, k = 1, p = 2, and q = 3, so that the two sections to be swapped are the ones in bold face in array b to the left below. Then b will be changed to the one on the right. In the example, every array element except the ones to be swapped is 0, so you can most easily see what the procedure does. Of course, these elements need not be 0.

```
      0 2 1 0 0                    0 3 4 0 0
b:    0 8 9 0 0     resulting b:   0 5 6 0 0
      0 0 0 3 4                    0 0 0 2 1
      0 0 0 5 6                    0 0 0 8 9
```

```
/** = Swap sections b[h..h+n–1][k..k+n–1] with b[p..p+n–1][q..q+n–1].
      Precondition: n ≥ 0,  the two sections do not overlap, and both sections are completely contained
                    in the array. There is no need to check for subscripts out of bounds. */
public static void swap(int[][] b, int n, int h, int k, int p, int q) {




}
```

Question 4 (12 points). Strings and recursion.

(a) Write the following function. You may not use a loop, and you must use recursion. As an example, `occ(5, 'b')` produces `'bbbbb'`.

```
/** = a String containing n occurrences of character c.  Precondition: n ≥ 0 */
public static String occ(int n, char c){




}
```

(b) Write the following function. Use recursion. Do not use loops. You may use calls on function `occ`, specified above. The only String functions needed are `charAt` and `substring(i)`.

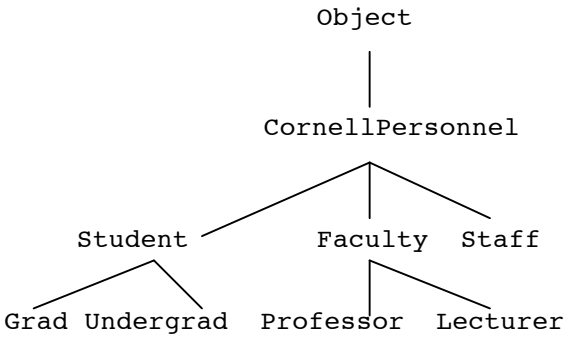
You will need to convert a character-digit like `'4'` to an **`int`**. Remembering that **`char`** values are represented as **`int`**s, we realize that `'4' – '0'` gives the **`int`** `4`.

```
/** Return s but with each pair "ic" of characters, where i is a digit, replaced by i occurrences of c.
    Example: the call p("2A0B3V")  produces a string with 2'A's, 0 'B's, and 3'V's, i.e. it produces
                                   "AAVVV".
  Precondition: s contains an even number of characters, and the first of each pair is a digit.  */
public static String p(String s) {




}
```

Question 5 (25 points). Questions on classes.
The questions on classes and objects deal with the hierarchy of classes shown to the right, which are used to maintain a database of people at Cornell. In addition, there are classes `Name`, `Address`, `Transcript`, `College`, `GraduateDegree` (e.g. MEng in CS, PhD in ChemE), and `Date`.



(a) Suppose the body of the single constructor in class `Student` does not start with a call on a superclass constructor. Write down the constructor call that Java inserts as the first statement in the body of the constructor.

(b) What are the steps in evaluating the new-expression `new Student("Doe", "Fall", 2006)`?

(c) Consider the following assignment:

```
CornellPersonnel s= new Student("Doe", "Fall", 2006);
```

Write down *all* classes to which variable `s` can be cast. State which of these casts can be done implicitly and which the programmer must program explicitly. Write down the Java code for one such explicit cast.

(d) After execution of the assignment in part (c), what are the apparent and real classes of variable `s`?

(e) Consider a variable `v`, with type `Vector<CornellPersonnel>`. Give an expression that tells whether or not element `v.get(i)` is of class `Faculty`. Give an expression, not a statement.

(f) Give a definition of Java keyword **`this`**. For example, what does it mean in a call like this?

```
m.add(this)          // here, m could be a Vector
```

(g) Below are 5 fields (or, rather, what should go in the fields) to contain information about Cornell personnel. Next to each, state the class in which you would declare the field and what its type would be. Before you answer, look at the classes mentioned at the beginning of this question 5.

a. The person's name.

b. The person's address.

c. Which college the person teaches in.

d. Which graduate degree program the person is enrolled in.

e. The person's transcript.

(h) Define "parameter" and "argument".

(i) Suppose `fi` is a **private** field of class `Faculty`. State where `fi` can be referenced (used).

(j) Write a method `equals`, to be placed in class `CornellPersonnel`, with this specification shown below. Assume the classes of the fields being compared have their own `equals` methods.

```
/** = "ob is a non-null object of class CornellPersonnel whose fields name and address
      are the same as this object" */
public Boolean equals(Object ob)
```

}

Question 6 (12 points). Part of classes `Lecturer` and `Faculty` mentioned in question 5 are shown below. Each `Lecturer` may have a mentor: a faculty member who looks after the lecturer, giving advice. A faculty member may be the mentor of several lecturers.

Four methods are shown in these classes: `makeMentor(f)`, `removeMentor()`, `addMentee(lec)`, and `removeMentee`(lec). A call to one of them **must** maintain the definition of both field `m` in objects of class `Faculty` and field `mentor` in objects of class `Lecturer`. For example, if method `f.removeMentee(lec)` is asked to remove `lec` as being mentored by `f`, this method should also be sure to make `lec.mentor` null (by calling `lec.removeMentor()`).

Methods in one class must call methods in the other. But an infinite set of calls should not happen, e.g. `lec.removeMentor()` calls `f.removeMentee(lec)`, which calls `lec.removeMentor()` again, which `calls f.removeMentee(lec)`, and on and on. *Some duplicate calling may be necessary*, but it should not be infinite.

Write the four method bodies. Do not write any loops; they are not necessary. Instead, use the `Vector` methods shown in the table at the right. Do not introduce other variables.

Below, `m` is a `Vector`.

`m.add(lec)`: add `lec` to `m`.

`m.remove(lec)`: remove `lec` from `m`.

`m.contains(lec)`: **true** if `m` contains `lec`; **false** otherwise.

```
public class Lecturer extends Faculty {

    // this lecturer's mentor (null if none)
    private Faculty mentor;

    /** Make f be this Lecturer's mentor (if f is
      already the mentor, there is nothing to do;
      if someone else is the mentor, first
      remove that mentor).
     Precondition: f is not null. */
    public void makeMentor(Faculty f) {









    }

    /** If this lecturer has a mentor, remove
        that mentor. */
    public void removeMentor() {









    }
}
```

```
public class Faculty extends CornellPersonnel {

    /** this faculty member is the mentor
        of lecturers in m. */
    private Vector<Lecturer> m=
                          new Vector<Lecturer>;

    /** If this faculty member is not lec's mentor
        make this faculty member lec's mentor.
       Precondition: lec is not null. */
    public void addMentee(Lecturer lec) {









    }

     /** Make sure that this faculty member is
         not lec's mentor --remove lec from this
         faculty member's list if necessary.
         Precondition: lec is not null. */
    public void removeMentee(Lecturer lec) {









    }
}
```

Question 7 (12 points). Matlab.
(a) Is the following true or false? Explain your answer.

The value of `5/3` is the same in Java and Matlab.

(b) Write Matlab code to compute an array that contains the first n partial sums of the following sequence. (The first element of the array will contain the first term, the second element the sum of the first two, the third element the sum of the first three, etc.). You may not use loops or recursion. Do not write a function; just write a sequence of assignments (if necessary) and a final expression.

$$\frac{1*2}{2*2} - \frac{2*3}{4*4} + \frac{3*4}{6*6} - \frac{4*5}{8*8} + \ldots$$