Grades for the final will be posted on the CMS as soon as it is graded, hopefully tonight but perhaps tomorrow. Grades for the course will take a few days more. You can look at your final when you return in the fall. HAVE A MERRY XMAS[1] AND A HAPPY NEW YEAR!

Please submit all requests for regrades for things other than the final BY 8PM TONIGHT. Use the CMS where possible; email Gries otherwise.

You have 2.5 hours to complete the questions in this exam, which are numbered 0..8. Please glance through the whole exam before starting. The exam is worth 100 points.

Question 0 (2 points). Print your name and **net id** at the top of each page. Please make them legible.

Question 1 (10 points) Known algorithms. Write a static function that implements the partition algorithm to partition a segment of an array, returning the index of the pivot value.

First, you *must* write the specification first, as a comment that precedes the header of the function. It must include: the pre-condition and postcondition, and an indication of what value is returned.

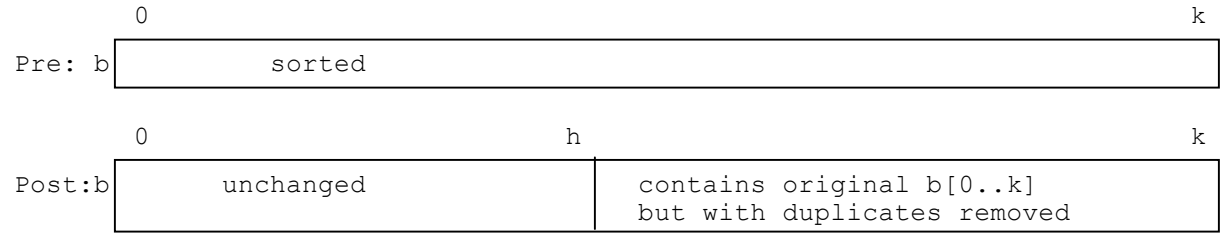
Question 0. _____	(out of 02)
Question 1. _____	(out of 10)
Question 2. _____	(out of 13)
Question 3. _____	(out of 12)
Question 4. _____	(out of 10)
Question 5. _____	(out of 14)
Question 6. _____	(out of 12)
Question 7. _____	(out of 15)
Question 8. _____	(out of 12)
Total _____	(out of 100)

Second, include all necessary parameters for the function to work properly.

Third, you *must* write an appropriate loop invariant, and your loop *must* be written according to the loop invariant.

[1] I like to say Xmas instead of Christmas. X is the mathematical variable, and the sayer and listener can, in their minds, substitute their own preference for it, and all will be happy —Christ, Buddha, Mohamed, Rama, Krishna, etc.

Question 2 (13 points) Loops. Array segment b[0..k] is already sorted (in ascending order), but it may contain duplicates. We want an algorithm that will remove the duplicates, as indicated by the following pre- and post-conditions.



For example, for the array b = {1, 2, 2, 2, 4, 4, 4}, execution of the algorithm sets h to k–3 (because there are 3 distinct values) and changes array b to this: b = {1, 2, 2, 2, 1, 2, 4} —the underlined part is the unchanged b[0..h], and the last three values are the elements of the original array b, without duplicates.

Write one loop (with initialization) that performs this task —we have provided the outline of the loop for you, below. You **must** use the invariant given below. Here are some points to consider.

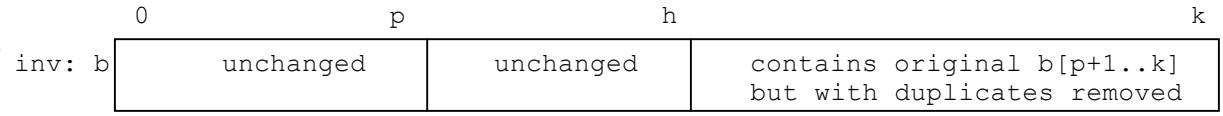
1. The invariant indicates that b[h+1..k] contains all the non-duplicated values in b[p+1..k], so that b[p] is the next value to check.

2. The repetend must determine whether b[p] is a duplicate –whether it already appears in b[h+1..k]. How can this test be done simply, using the fact that b is sorted?

// 0 <= k (the segment to be processed has at least one element)

p= _____;

h= _____;

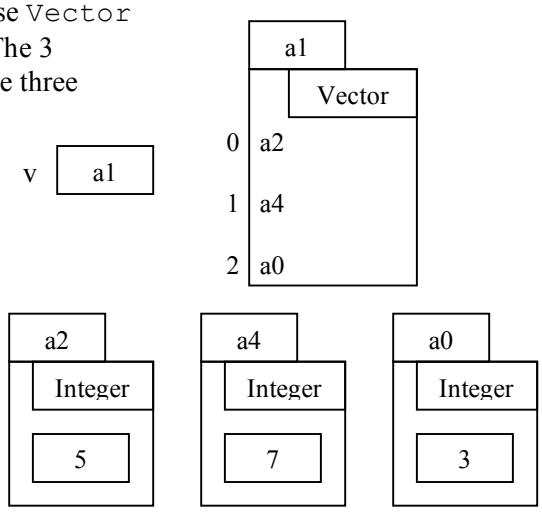



```
while (                    ) {




}
```

Question 3 (12 points) Executing method calls. Suppose `Vector` v of Integers contains 3 elements, as shown to the right. The 3 elements are the names of `Integer` objects wrapping the three **`int`** values 5, 7, and 3.



Execute the method call

```
VectorTools.reverse(v);
```

where class `VectorTools` is given below. (We have labeled the statements with numbers (e.g. 2:), which you can use as program counters in a frame for a call). Stop executing when you are ready to execute the return statement that is labeled 2.

For each call during execution (except the calls on methods in class `Vector`), draw the frame for the call. Note that elements of `Vector v` will change, and you should record those changes in object `a1`.

Hint: You will have to draw 2, 3, or 4 frames for calls.

Note: For v a Vector,
v.size() = number of elements in v.
v.get(i) = the value of element v[i]
v.set(i,w) sets v[i] to w.

```
public class VectorTools {
    // Reverse v[0..v.size()]
    public static void reverse(Vector v) {
        1: reverse(v, 0, v.size()–1);
    }

    // Reverse the segment v[h..k]
    private static void reverse(Vector v, int h, int k) {
        1: if (h >= k)
            2: return;

        3: Integer w= v.get(h);
        4: v.set(h, v.get(k);
        5: v.set(k, w);
        6: reverse(v, h+1, k–1);
    }
}
```

Question 4 (10 points) Exception handling.

(a) Consider class ExException, shown to the right. If the following statement legal? If not, explain why.

```
throw new ExException();
```

```
public class ExException {
   public ExException() {
         super();
   }
}
```

(b) In the box to the right is a class C, which contains two static methods. Below, we give the "interactions pane" of DrJava, with a method call in it.

Execute the method call and put in the interactions pane the output of all the println statements that are executed.

You need not draw frames for method calls, if you yourself do not feel the need to do so. Just make sure to put the right information in the interactions pane.

```
> C.m1(5);
```

```
public class C {
   public static void m1(int p1) {
      try {
         System.out.println("1: " + p1);
         p1= m2(p1 + 1);
         System.out.println("2: " + p1);
      } catch (ArithmeticException e) {
         System.out.println("3: " + p1);
      }
      System.out.println("4: " + p1);
   }

   public static int m2(int q1) {
      q1= q1 + 1;
      try {
         System.out.println("5: " + q1);
         if (q1 != 10)
            throw new  IOException();
         System.out.println("6: " + q1);
      } catch (ArithmeticException e) {
         System.out.println("7: " + q1);
         q1= q1 / 0;
         System.out.println("8: " + q1);
      } catch (IOException e) {
         System.out.println("9: " + q1);
         q1= q1 / 0;
         System.out.println("10: " + q1);
      }
      System.out.println("11: " + q1);
      return q1+2;
   }
}
```

Question 5 (14 points). Recursion. Write the body of the recursive method specified below. It stores in a square t by t array segment the integers n, n+1, n+2, … in spiral fashion. We show below the results for t = 2, 3, 4, 5, all starting with n = 0. As you can see, one fills the upper row with successive integers, then the right column, then the lower row (in reverse order), then the left column (in reverse order), and finally the middle square in the same fashion. The complete spec is given below the examples.

0 1	0 1 2	00 01 02 03	00 01 02 03 04
3 2	7 8 3	11 12 13 04	15 16 17 18 05
	6 5 4	10 15 14 05	14 23 24 19 06
		09 08 07 06	13 22 21 20 07
			12 11 10 09 08

To help you out, use the two functions specified to the right can to fill in a segment of a row or column. Naturally, if r1 > r2 or c1 > c2, nothing is changed.

As an example,

fillRow(b, 0, 2, 3, true, 20)

stores 20 in b[0][2], stores 21 in b[0][3], and returns the value 22.

You may not use a loop, and you do not have to write any other method.

```
/** Store integers n, n+1, ... into b[r][c1..c2] of row r
    and return the first integer that is not used. If
    forward is true, fill them in the order c1, c1+1,
    ..., c2; if false, in the order c2, c2-1, ..., c1*/
public static int fillRow(int[][] b, int r, int c1, int c2,
                          boolean forward, int n)

/** Store integers n, n+1, ... into b[r1..r2][c] of column
    c and return the first integer that is not used. If
    forward is true, fill them in the order r1, r1+1,
    ..., r2; if false, in the order r2, r2-1, ..., r1*/
public static int fillCol(int[][] b, int r1, int r2, int c,
                          boolean forward, int n)
```

```
/** Fill in the elements of b[h..k][h..k] with integers n, n+1, n+2, ... in spiral fasihion , as
    shown by the 4 examples given above –filling in the outer rows and columns first
    and then filling in the middle of b[h..k][h..k] in the same fashion. */
public static void spiral(int[][] b, int h, int k, int n) {




}
```

Question 6 (12 points). The purpose of this question is to test your ability to deal with programs that have static variables and that have a bit of aliasing. Below is a class `Game` with 4 methods: constructor `Game`, procedure `changePts`, function `toString`, and procedure `createGames`. Execute the call

Game.createGames();

Use the empty box below to write the output of the println statements, or use the back of the previous page if you wish. In executing the call, we encourage you to first draw all the local variables of `createGames` and to draw any objects that are created. You do not have to draw frames for method calls.

```
public class Game {
  private static int totalPts= 0;

  private String team1;
  private String team2;
  private int pts;

  public Game(String t1, String t2, int p) {
    team1= t1;
    team2= t2;
    pts= p;
    totalPts= totalPts + p;
  }

  public void changePts(int p) {
    totalPts= totalPts – pts + p;
    pts= p;
  }

  public String toString() {
    return team1 + " vs " + team2 + ": " +
           pts + " pts. Total: " + totalPts;
  }
```

```
  public static void createGames() {
    totalPts= 0;
    Game g1= new Game("Knicks", "Nets", 100);
    System.out.println(g1);
    Game g2= new Game("Knicks", "Nets", 120);
    System.out.println(g2);
    Game g3= new Game("Knicks", "Nets", 120);
    System.out.println(g3);

    System.out.println("g1 = g2: " + (g1 == g2));
    System.out.println("g2 = g3: " + (g2 == g3));
    Game g4= g2;
    g1= g2;
    g2= g3;

    g1.changePts(80);
    System.out.println(g1);
    System.out.println(g2);
    System.out.println(g3);
    System.out.println(g4);

    System.out.println("g1 = g2: " + (g1 == g2));
    System.out.println("g1 = g4: " + (g1 == g4));
  }
}
```

Question 7 (15 points) Classes. Part of classes `Course` and `Instructor` mentioned are shown below. Each `Course` may have an `Instructor`, and an instructor may teach several courses.

Four methods are shown in these classes: `makeInstructor(t)`, `removeInstructor()`, `addCourse(c)`, and `removeCourse(c)`. **A call to one of them must maintain the definition of field `courses` in all objects of class `Instructor` and also field `person` in all objects of class `Course`.** For example, if method `f.removeCourse(c)` is asked to remove course `c`, this method should also be sure to make `c.person` null (by calling `c.removeInstructor()`).

Methods in one class must call methods in the other. But an infinite set of calls should not happen, e.g. `c.removeInstructor()` calls `person.removeCourse(c)`, which calls `c.removeInstructor()`, which calls `person.removeCourse(c)`, 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.

You may introduce local variables but no fields or static variables. Do not write any other methods.

Below, `m` is a `Vector`.

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

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

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

```
public class Course {
  // the instructor teaching this course (null if none)
  private Instructor person;

  /** Make t be this course's instructor (if t is
      already the instructor, there is nothing to do;
      if someone else is the instructor, first
      remove that instructor).
      Precondition: t is not null. */
  public void makeInstructor(Instructor t) {









  }

  /** If this course has an instructor, remove
      that instructor. */
  public void removeInstructor() {









  }
}
```

```
public class Instructor {
  /** this instructor is teaching classes in c. */
  private Vector<Course> courses= …;

  /** If this instructor is not teaching course c,
      add this as a course it is teaching.
      Precondition: c is not null. */
  public void addCourse(Course c) {









  }
  /** Remove c from this
      instructor's course list, if necessary.
      Precondition: c is not null. */
  public void removeCourse(Course c) {









  }
}
```

Question 8 (10 points) Miscellaneous.

(a) Name the layout managers associated with objects of class JFrame and Box and explain how components are layed out with each of them.

(b) Write the four loopy questions for understanding a loop.

(c) In Java, what is a wrapper class? What two purposes do they serve?

(d) In assignment A8, procedure `hide` did its job quickly, no matter how long the message being hidden. But procedure `reveal` took perceptibly more time the longer the message got —even minutes for a message of 5,000 characters. We discussed this issue and decided that it was because of the statement

```
s= s + character;
```

which, each time it was executed, appended the next character to `String s`, so that when all the revealed characters had been appended, s contained the message. Discuss briefly what happens when this statement is executed and why its repeated execution might take a long time when revealing a 5,000 character message.