Question 0.	_________	(out of 01)
Question 1.	_________	(out of 10)
Question 2.	_________	(out of 12)
Question 3.	_________	(out of 12)
Question 4.	_________	(out of 12)
Question 5.	_________	(out of 12)
Question 6.	_________	(out of 16)
Question 7.	_________	(out of 15)
Question 8.	_________	(out of 10)
Total	___________	(out of 100)

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 NICE SUMMER!

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

Question 1 (10 points). Matlab. Write Matlab code, with no loops and no recursion, to compute the cumulative sum of `n` approximations of the Taylor series given below to find the value of `ln(6)`. Do not write a complete function. Just write the sequence of statements and final expression that gives the result. Note: `ln` is the natural logarithm. You do not need to know what `ln` or Taylor series mean to solve this problem.

$$\ln(1 + x) = x/1 - x^2/2 + x^3/3 - x^4/4 + x^5/5 - \ldots$$

Question 2 (12 points). Loops. One can sum the digits of a nonnegative integer, like 563, giving 14, and then sum again, giving 5. 5 is called the *digit sum* of 563. Another example: the digit sum of 74 is calculated in two steps: 74 → 11 → 2.

When calculating the digit sum, one can do it as follows, always keeping the answer less than 10:

```
     (digit sum of 867)
=    (digit sum of 86) + 7
=    (digit sum of 8) + 4      [note: 6 + 7 = 13; we summed 1 and 3 to get 4]
=    (digit sum of 0) + 3      [note: 8 + 4 = 12; we summed 1 and 2 to get 3]
=    3
```

Complete the program segment below, which stores the digit sum of `n` in `c`. The postcondition and invariant of the loop is given. Your code should be consistent with the invariant —remember the four loopy questions. Any new variables you introduce must be declared in the repetend, not outside the loop. Operations …/10 and …%10 may come in handy.

```
int m=                    ;


int c=                    ;
// invariant: (digit sum of n)  =  c + (digit sum of m)   and  0 <= c < 10  and  0 <= m
// Termination: m will decrease at each iteration
while (                          ) {











}
// postcondition:  (digit sum of n)  =  c
```

Question 3 (12 points). Arrays and methods. Array `scores` contains a list of scores, all in the range `0..20`. Write the body of function `histo` below, which makes a histogram in a String array `s` (say). For example, if `scores = {2,0,2,2,0,0,0,3,0,3,0,4,3,0,2}` then `s[0..5]` is the following:

```
"00 *******"
"01 "
"02 ****"
"03 ***"
"04 *"
"05 "
```

and the rest of the array elements of `s` contain no `"*"`s because no element in `score` is over 4.

Hint. We suggest first creating a local array `b`, where each `b[i]` contains the number of times `i` occurs in array `scores`.

/** = Precondition. Array `scores` contains integers in the range 0..20.
Return an array `s` (say) of Strings that gives a histogram of `scores`. Each element `s[i]` contains:

(1) `i`, as a two-digit integer (with leading zeros if necessary),
(2) a blank,
(3) `n` asterisks "*", where `n` is the number of times `i` appears in array `scores`.
*/

```
public static String[] histo(int[] scores) {











}
```

Question 4 (12 points). On this page are class definitions for classes `Sound` and `Song`. On the next page is a class `TABand`. Execute this call:

```
TABand.recordingSession();
```

Write the output of println statements directly beneath the println statements, in the space provided (on the next page). If an error would be the result of the statement write ERROR.

We suggest that you start by drawing all local variables of method `recordingSession`. Then, as you execute the sequence, draw all objects created and execute any assignment statement faithfully. You don't have to do this, but you will have a better chance of getting correct answers if you do.

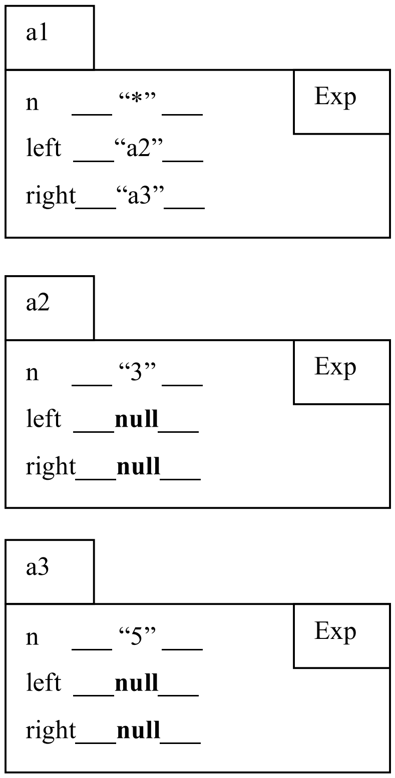
```
public class Sound {
  /* Total duration of all instances of Sound */
  private static double totalDuration=  0.0;

  private double duration; // duration of sound

  /** Constructor: new Sound with duration m*/
  public Sound( double m) {
     duration=  m;
     totalDuration=  totalDuration + duration;
  }
/** = duration of this sound */
  public double getDuration()
     { return duration; }

  /** = the total duration of all sounds */
  public static double getTotalDuration()
     { return totalDuration; }

  /**Rreplace sound with new one of duration m */
  public void overwrite(double m) {
     totalDuration =  totalDuration – duration + m;
     duration =  m;
  }

  /** Append to this sound a new sound of
       duration m */
  public void append(double m) {
       duration=  duration + m;
       totalDuration=  totalDuration  + m;
  }
}
```

```
public class Song extends Sound{
  /* total number of instances of Song created */
  private static int numSongs=  0;

  private String name; // name of song
  private String author; // author of song

  /** Constructor: a new Song with name t,
                  author s, and duration m*/
  public Song(String t, String s, double m) {
     super(m);
     name=  t;      author =  s;
     numSongs=  numSongs + 1;
  }

  /** = "<name> - <author> : <duration>" */
  public String toString(){
     return name + " - " + author + " : " +
               getDuration();
  }

  // = the author of this song
  public String getAuthor()
     { return author;  }
}
```

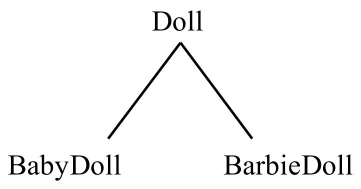
```
public class TABand {
  public static void recordingSession() {
      Sound first= new Song("CS100 Blues", new String("Gries"), 6);
      Song forth= new Song("Help!", "Student", 12);
      Sound wail= new Sound(30);
      Song fifth= new Song("I will teach you", new String("Gries"), 10);
      Sound humming= forth;
      Song second= (Song) first;
      humming.overwrite(10);
      first.append(3.0);
      System.out.println(first);


      System.out.println(second);


      System.out.println(forth);


      System.out.println(wail);


      System.out.println(humming);


      System.out.println("Length of sound: " + Sound.getTotalDuration());


      System.out.println("The author is: " + second.getAuthor());


      System.out.println("The author is: " + first.getAuthor());


      System.out.println("The songs have the same author?" +
               (second.getAuthor() == fifth.getAuthor()));


      System.out.println("The songs have the same author?" +
               (second.getAuthor().equals(fifth.getAuthor())));


  }
}
```

Question 5 (12 points). Recursion. Class Exp, given below, is used to represent expressions. For example we show to the right below three objects of class Exp that are used to represent the expression 3 * 5. And below them are two "trees" that represent expressions. Write the body of (recursive) function Exp.eval, which is in class Exp. Do not use loops; use recursion. To change a String value s to an integer, you may use function Integer.parseInt(String s).

```
/** An instance represents an expression constructed
    from integers and the infix operators +, –, and *. */
public class Exp {
   String n;        // a (possibly negative) integer or an
                    // operator "=", "-", or "*"
   Exp left;        // null if n is an integer; otherwise the left operand
   Exp right;       // null if n is an integer; otherwise the right operand

   /** Constructor: an expression consisting of the integer i */
   public Exp(int i) {
      n= "" + i;
   }

   /** Constructor: an expression with operator op
                  (one of "+", "-", and "*"), a left
                  operand left, and a right operand right. */
   public Exp(String op, Exp left, Exp right) {
      n=  op;
      this.left=  left;
      this.right=  right;
   }

   /** = value of expression e. */
   public static int eval(Exp e) {




   }
}
```



Question 6 (16 points). Classes. An online retail store maintains information about the dolls they sell. They classify the dolls as shown in the figure to the right. Class `Doll`, shown to the right below, is used to represent dolls. Every `Doll` has a name, price, and seller name



(a) A `BabyDoll` can optionally laugh. Class `BabyDoll` is defined (partially) below. Your task it to complete the body of the constructor and function equals.

(b) After executing the following two statements what will be the apparent and real classes of variable d? [2 points]

```
BabyDoll b= new BabyDoll (“Edeline”, 100, “eToys”, true);
Doll d= b;
```

```
public class BabyDoll extends Doll{
  boolean canLaugh;  // = “doll can laugh”

  /** Constr: BabyDoll with name n, price p, and seller s.
      Parameter b indicates whether this doll can laugh */
  public BabyDoll(String n, double p,
                  String s, boolean b){









  }

  /** = “ ob is an instance of class BabyDoll and has the same
         values in its fields as this BabyDoll” */
  public Boolean equals(BabyDoll ob) {

















  }
}
```

```
public class Doll {
   private String name;  // Doll’s name
   private double price ;// Doll’s price
   private String seller; //  Doll’s seller

   /** Constructor: Doll with name n
                      price p, and seller s */
   public Doll(String n, double p,
                            String s) {
      name= n;   price= p;  seller= s;
   }
   /** = name of doll */
   public String getName()
      {  return name; }
   /** = price of the doll */

   public double getPrice()
      { return price; }

   /** = seller’s name */
   public String getSeller(){
      { return seller; }

   /**  = representation of Doll*/
   public String toString(){
      return "Name: " + getName() +
             ", Price: " + getPrice() +
             ", Seller: " + getSeller();
   }
}
```

(c) Below, write a definition for class `BarbieDoll`, which should have the following properties.

1. It extends class Doll.
2. It has a String field that the color of the BarbieDoll's hair.
3. It has a String field that gives the style of the BarbieDoll's clothes.
4. It has a constructor, which allows the caller to give values for name, price, seller, hair color, and style.
5. It has getter methods for the two fields.
6. It has a function toString that is similar to that in class Doll except that it also gives the values of the two fields in this class. The information about the inherited fields should be obtained by calling the inherited toString function.

Question 7 (15 points). Miscellaneous.
(a) What does function `equals` in class `Object` do?.

(b) What constructor call is inserted in a constructor for a subclass if the first statement is not a constructor call?

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

```
m.add(this);
```

(d) To the right is a definition of a class `Qd.`

First, draw the file drawer for Qd just before execution starts.

Second, write down the steps in evalating the expression **new** Qd(3, 4). With each, actually do the step yourself, drawing any objects and calls for frames that are required, putting the objects where they belong (place frames for calls anywhere on the page).

```
public class Qd {
   public static int s= 5;
   public int f;
   public Qd(int f, int t) {
      this.f= f;
      s= s + t;
   }
}
```

If you have to execute a method call, draw the frame for the call, but you do not have to explain the steps in executing the method call.

Question 8 (10 points). Algorithms.

Write algorithm `Partition` as a function, complete with method header (giving the parameters, for example) and a suitable specification —a precondition and postcondition. You may give the pre- and post-conditions as formulas or as pictures. 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.

Requirements. The function must partition array segment b[p..q–1]. Your answer must give a suitable invariant and the loop must be developed from the invariant.

Because many of you have not been reading carefully on exams, we give you these hints. After you have finished writing your answer, go back and read the preceding paragraph again and think about the following.

1. Did you start out by writing a specification (precondition and postcondition) and header for the function? Does your specification say what value is returned?

2. Does your function partition array segment b[p..q–1]?

3. Did you write a suitable invariant before writing the loop, and does your loop conform to the invariant?

4. Did you write a return statement?