This sample solution corresponds to version A of the test. Other versions had slightly different questions or had questions in a slightly different order.

Several questions asked you to design an algorithm or class. There are many possible answers, not just the ones shown here, and reasonable answers received full credit.

Question 1. (8 points) You are in a kitchen where the following items are on the counter: a sliced loaf of rye bread, a bottle of mustard, a 2-lb. block of swiss cheese, and an unopened bag of turkey coldcuts. Give an algorithm for making a turkey and swiss cheese sandwich on rye bread with mustard. Give specific instructions describing each step. The instructions should be detailed enough so that a person who has never made a sandwich before will get something close to the desired result. (In other words: more detail than "grab some bread and make a sandwich")

- **Place two slices of bread on the table**
- **Open bag containing turkey**
- **Place pieces of turkey on one piece of bread**
- **Cut a slice of cheese from the block**
- **Place the slice of cheese on top of the turkey**
- **Spread some mustard on the other slice of bread**
- **Place the slice of bread with the mustard on top of the other slice with the turkey and cheese**

Question 2. (5 points) Assume that the following assignment statements have been executed.

```
int x = 17;
int y = 42;
int z = -12;
boolean isReady = true;
```

For each of the following boolean expressions, indicate whether the expression is `true` or `false` by circling the correct answer.

(a) true [✓ false] `x >= z && x >= y`

(b) true [✓ false] `!(x == y || y > z)`

(c) true [✓ false] `isReady && (z > 0)`

(d) [✓ true] false `((x + y) > 50) && ((x / y) == 0)`

(e) [✓ true] false `!isReady || (z <= 0)`

Question 3. (8 points) Suppose we are designing a computer system for an online television program listing that can be used to find out what programs are available on different channels throughout the day.

(a) One kind of object in this system might be a `TVShow` object describing a single television program. List three properties and their types and three responsibilities that might be appropriate for a `TVShow` object. For each responsibility indicate whether it is a command or a query.

Property	Type
title	**String**
startTime	**Time (or int or double)**
length	**int**
channel	**int**

(many other possibilities)

Responsibility	Command or Query?
getTime	**query**
getTitle	**query**
changeTime	**command**

(b) Describe a second kind of object that might also be appropriate for this system. List three properties and their types and three responsibilities that might be appropriate for an object of this class.

Name of second kind of object ___**ChannelSchedule**_____________________.

Property	Type
programs	**list of TVShows**
channel	**int**
date	**Date**

Responsibility	Command or Query
addProgram	**command**
getProgramAt(time)	**query**
getAllPrograms	**query**

Question 4. (5 points) In lecture we developed a HuskyCard class. Here is some of the code from that class, with comments and several methods removed to save space.

```
public class HuskyCard {
   private String name;
   private int id;
   private int balance;

   public HuskyCard(String ownerName, int idNumber) {
      name = ownerName;
      id = idNumber;
      balance = 0;
   }
   public int getBalance() {
      return balance;
   }
   public void deposit(int amount) {
      balance = balance + amount;
   }
   public boolean withdraw(int amount) {
      if (amount <= balance) {
         balance = balance - amount;
         return true;
      } else {
         return false;
      }
   }
}
```

Choose words or phrases from the following list **only** to *best* complete the sentences below.

assignment, boolean, class, class definition, client, constructor, declaration, double, expression, integer, instance variable, instance, local variable, loop, message, method, name thingy, parameter, precedence, return statement, return value, scratch space, state, String, this, type, void

In the definition of class `HuskyCard`, the line of code reading "`private String name;`" is the declaration of a/an **instance variable** of the class. In the definition of deposit, the identifier amount is a/an **parameter** of that method. In method `withdraw`, the **boolean** expression `amount <= balance` is used to check whether the user is trying to withdraw more money than is available. The line of code that begins with "`public HuskyCard(…)`" defines the **constructor** of this class. The **type** of the return value of method `getBalance` is `int`.

Question 5. (8 points) In class we had several students play the role of different kinds of Performer objects. One of these was an Actor. Actors, as well as all other kinds of Performers, had three responsibilities: they could twirl *n* times, clap *n* times, and, in response to a `tellCount` message, report the total number of claps and twirls they have done.

For this question, provide a Java specification (***not*** implementation) for a class `Actor` that can carry out these responsibilities. You should include appropriate, but brief and succinct, JavaDoc comments, and include headings for the appropriate methods and/or constructors.

Reminder: In JavaDoc, @param and @return are used to document method parameters and return values respectively.

```
/** Specification of an Actor */
public class Actor {

   /** Have this performer clap
    * @param n  number of times to clap
    */
   public void clap(int n) { ... }

   /** Have this performer twirl around
    * @param n  number of times to twirl
    */
   public void twirl(int n) { ... }

   /** Report the number of twirls and claps
    * @return total number of twirls and claps this Actor has done
    */
   public int tellCount() { ... }
}
```

Question 6. (8 points) The following code specifies a class `Clock` that keeps track of a ticking 12-hour clock. Each time the clock ticks, the time should be advanced by 1 minute. If a tick occurs when the clock is recording 59 minutes, then the minutes should be reset to 0 and 1 should be added to the hour. If the hour is already 12, advancing it by 1 hour should set it to 1. You should provide implementations of the `Clock` constructor and `tick()` method below. Use the hours and minutes instance variables as defined. You can add additional variables if you need them.

```
/** Simple 12-hour clock */
public class Clock {
   // instance variables
   private int hours;              // hours from 1-12
   private int minutes;            // minutes from 0-59



   /** return the clock's current value for hours
    * @return current hours from 1-12 */
   public int getHours() {
      return hours;
   }

   /** return the clock's current value for minutes
    * @return current minutes from 0-59 */
   public int getMinutes() {
      return minutes;
   }

   /** Construct a new clock set the time to 12:00 */
   public Clock() {

      hours = 12;
      minutes = 0;

   }

   /** Advance the time by 1 minute */
   public void tick() {

      minutes = minutes + 1;      // or minutes++;
      if (minutes > 59) {
         minutes = 0;
         hours = hours + 1;
         if (hours > 12) {
            hours = 1;
         }
      }

   }
}
```

Question 7. (8 points) Trace through the following code and show how the value of n changes as the code executes: i.e., to the right of the phrase "values of n", whenever a new value is assigned to n, cross out the previous value and write the new one to the right. What output is produced when the code is executed?

```
int n = 6;
while (n > 0) {
   System.out.println(n);
   n = n - 2;
}
```

values of n: **~~6 4 2~~ 0**

output (it's ok to write all the numbers on a single line, clearly separated):

6 4 2

(b) Complete the following loop so it calculates the sum 2 + 5 + 8 + 11 + 14 + 17. There should only be one addition to variable sum on each iteration of the for loop.

```
int sum = 0;

for ( int n = 2 ; n <= 17 ; n = n + 3 ) {

   sum = sum + n ;

}
```