CSCI 131, Last Exam – Review Questions

This sheet is intended to help you prepare for the last exam in this course. The exam will cover all topics in the course, but there will be more emphasis on the topics covered since exam two, i.e. material from chapters 2.3, 3.1, 3.2, and parts of 3.3. The focus is on new material (classes, objects, and recursion), but older material from previous exams (functions, loops, files, printing, etc.) will certainly be included. You may want to review topics from exams 1 and 2, since it is assumed you have mastered all earlier material, including loops, file input/output, conditionals, arrays, functions, and strings. Output formatting (i.e. printf) will not be emphasized. The following is a partial list of topics that are likely to be emphasized on the exam:

1. Using classes and Libraries
   Be able to call functions from Math, StdRandom, StdStats, and other similar libraries.
   Be able to read and understand documentation for a library, e.g. function headers.

2. ADTs: Using Existing Objects in Java
   String data type and String methods.
   Given documentation for a class (e.g. Star), be able to construct new objects and call methods.

3. Classes: Defining new types of objects
   Defining a new class
   Define and use instance variables (aka "member variables")
   Define and use constructors
   Define and use instance methods (aka "member methods" or "non-static methods")
   Related keywords: *this, super, final, public, private, protected*
   Object-Oriented Programming concepts and terminology
   Client code for creating and using objects
   Inheritance and the *extends* keyword

4. Recursion
   Definition of a recursive function
   Base Case vs. General Case
   Trace of recursive function (i.e. visualizing with a table)
   Invocation tree for recursive function (i.e. visualizing with a diagram)
   Using and writing recursive functions.

5. Static methods (aka functions)
   Writing and invoking functions, Function signatures/prototypes, Using arrays as parameters or return values, Call by value behavior for primitive types, reference behavior for arrays and objects, Function-local variables and variable scope.

Some questions have been taken directly from previous exams.

Practice Problems

The following problems are intended to help you study for the exam, however topics not covered here may be on the exam as well. Use your text, class notes, labs, assignments, and the internet as well. The Sedgewick & Wayne book site has many good exercises as well.

1) Examine the following static method and answer the questions below.

```
public static int calc(int a, int b) {
      if (a == 0) {
            StdOut.println ("Hurray!");
            return b;
      } else {
            StdOut.println (a + ", " + b);
            int x = calc(a - 2, b * 2);
            StdOut.println ("Result is " + x);
            return x + 1;
      }
}
```

a) What output is printed to the terminal during the following function call?

```
int getValue = calc(8, 3);
```

b) Are there any positive numbers for the parameters of this function that will cause infinite recursion? If so, give an example call. If not, explain why not.

2) (a) Define a class Country to represent information about literacy rates in each country. Each Country object should contain a name (e.g. "Egypt"), a population in millions (e.g. 91.5), and the numbers of people who are literate in that country in millions (e.g. 69.16). Give the class appropriate member variables, methods, and constructors so the following client code will work:

```
Country ind  = new Country("India", 1352.6, 975.2);
Country egy = new Country ("Egypt", 91.5, 69.17);
Country fra = new Country ("France",67.4, 67.2);
StdOut.println (ind + " has a literacy rate of % " + ind.getRate());
StdOut.println (egy + " has a literacy rate of % " + egy.getRate());
```

The effect of this code would be to print (don't worry about the number of decimals printed):

```
India has a literacy rate of % 72.1
Egypt has a literacy rate of % 75.6
```

(b) Add a member method hasHigherLiteracy() that compares this Country's literacy rate to another country's literacy rate. It should take one parameter, the other Country this one should be compared to, and it should return true if this country's literacy rate is higher than the other country's literacy rate.

(c) Suppose an array A of Country objects has already been declared and completely filled with initialized Country objects. Write client code that finds and prints the country that has highest literacy rate in the world. You can assume the array contains at least one country.

3) What is the result of calling `Mixer(5, 3)` ?

```
public static int Mixer(int x, int y) {
     if (x == 1)
        return y;
    else if (y == 1)
        return x;
    else
        return Mixer(x-1, y-1) + x + y - 1;

}
```

Draw a diagram with arrows and a box for each function call showing all the parameters, calculations, and return value for that function call. Or draw a table with one row for each function call, and columns showing the parameters, calculations, and return value for each row.

4) Consider this recursive function:

```
public static int subtract(int a, int b) {
      StdOut.printf(" %5d %5d \n", a, b);
      if (a > b)
            return subtract(a-b, b);
      else if (b > a)
            return subtract(a, b-a);
      StdOut.println("Exit");
      return a;
}
```

(a) What output is printed to the console during the following function call?

StdOut.println("Answer is " + subtract(63, 36));

(b) Are there any integers for the parameters of this function that will cause infinite recursion? If so, give an example call. If not, explain why not

5) Consider the following class definition (with the bodies of the constructors and methods omitted):

```
public class Student {
   private String name;
   private int id;
   private double gpa;

   public Student( ) { /* code omitted */ }
   public Student(String studentName, int studentID) { /* code omitted */ }

   public void changeName(String newName) { /* code omitted */ }
   public void changeID(int newID) { /* code omitted */ }
   public void changeGPA(double newGPA) { /* code omitted */ }
   public double getGPA() { /* code omitted */ }
   public String toString() { /* code omitted */ }
   public void print() { /* code omitted */ }
}
```

a) What are the member variables for this class? How many constructors are there? How many methods does it have?

b) Write code for a client to create a new Student object to represent a student named Francis, who has a GPA of 3.95 and who's ID number is 123456. Hint: there are 2 different ways to do this.

c) Write a new static function (not a member method in this class) named *computeAverageGPA*. This function should take as a parameter a completely-filled array of Student objects. It should compute the average GPA for those students, and return the result.

d) Use the function from (c) to write client code to print information about all the above-average students. You can assume there is already an array, *seniors*, that is completely filled with Student objects that have been initialized. Hint: there are 2 different ways to do the printing.

e) Complete the implementation of the toString( ) function for the Student class. This function should return a string that contains a description of the student, like "Student named Francis, with gpa 3.95 and ID 123456". Don't worry about the decimal places. Hint: Do NOT print this string... return it instead.

f) Complete the implementation of as many of the other methods and constructors as you can. For the constructors, unless otherwise specified, you should initialize all member variables to some default blank/zero/empty value.

6) Consider the following class definitions (with the bodies of the constructors and methods omitted):

```
public class Student {
   private String name;
   private int id;
   private double gpa;

   public Student( ) { /* code omitted */ }
   public Student(String initName, int initId) { /* code omitted */ }

   public void changeName(String newName) { /* code omitted */ }
   public void changeID(int newID) { /* code omitted */ }
   public void changeGPA(double newGPA) { /* code omitted */ }
   public double getGPA() { /* code omitted */ }
   public String toString() { /* code omitted */ }
   public void print() { /* code omitted */ }
}
public class GradStudent extends Student {
   private String advisorName;

   public GradStudent( ) { /* code omitted */ }
   public GradStudent(String initName, int initid, String initAdvisor) {
                                                   /* code omitted */ }

   public void changeAdvisors(String newAdvisor) { /* code omitted */ }
   public void print() { /* code omitted */ }
}
```

a) Which class is the parent (base) class and which is the child (derived) class?

b) Which methods are the same for both the Student and the GradStudent class?

c) What are the member variables for each class?

d) Which class methods and variables can be accessed directly by the member functions of each class?

e) The line "public GradStudent() …" is different from the other methods listed. How so? What is it for?

f) Complete the implementation of the print( ) function for the Student class. This function should print to the standard output the values of name, id and gpa of the Student object. Use a reasonably pretty format, such as "Student named Alice, with gpa 3.95 and ID 123456".

g) Complete the implementation of the print( ) function for the GradStudent class. This function should write to the standard output the values of name, id, gpa and year of the GradStudent object. Again, use a reasonably pretty format, such as:

Student named Ada, with gpa 3.9 and ID 24576
Student advisor is named Charles

h) Complete the implementation of GradStudent(String initName, int initId, String initAdvisor). This should initialize all the member variables appropriately. The GPA can be initialized to zero.

7) a) Define a class named `Point` to represent a point on the x, y plane, where both coordinates are floating point numbers. Give the class appropriate member variables, methods, and constructors, so that the following client code would work:

```
Point e = new Point(78.0, 50.5);
double d = e.distanceFromOrigin();
StdOut.println(e + " is about " + d  + " inches from the origin");
```

The effect of this code would be to print:

"Point at 78.0, 50.5 is about 96.9 inches from the origin"

The `distanceFromOrigin()` method should calculate and return the distance of this point from the origin, that is, sqrt($x^2 + y^2$).

(Hint: you also need a toString() method that returns a String like "Point at 78.0, 50.5".)

b) Write client code to ask the user to input two x, y pairs, then use two Point objects to check which point is further from the origin. Print the further point.

c) Suppose A is an array of Point objects that has already been initialized. Write code to find and print the point in the array that is closest to the origin.

d) Add a new member method named `move()`. It should take two floating point parameters, representing a horizontal distance and a vertical distance, and it should return nothing. The function should change the point so that it moves the specified horizontal and vertical distances. The goal is to allow this client code to work:

```
Point a = new Point(3.0, 5.0);
a.move(1.0, 6.0);
StdOut.println( a );
```

The effect of this code would be to print:

"Point at 4.0, 11.0"

e) Add new member method named `moveTowards()`. It should take one parameter, a Point object named T, and it should return nothing. The function should change this point so that it moves half-way towards the point T. For example, if client code does this:

```
Point a = new Point(3.0, 5.0);
Point b = new Point(1.0, 10.0);
a.moveTowards(b);
StdOut.println( a );
```

The effect of this code would be to print:

"Point at 2.0, 7.5"

since that is the half-way point between a's original position and b.

f) Write a static function named `centerOfGravity()`. It should take an array of Point objects as a parameter and return a Point object as a result. The function should calculate the center of gravity of all the points in the array and return a new point at that location. Note: the x, y coordinates for the center of gravity is just the average of all of the x and y coordinates of the points. If you need to add new member methods to the Point class, then do so, but otherwise do not modify the Point class.

8) This is a variation of question #7, but now using inheritance. Suppose we have created a class `Point` to represent a point on the x, y plane, where both coordinates are floating point numbers:

```
public class Point {
      protected double x, y;
      public Point(double xx, double yy)  {  /* code omitted */  }
      public String toString()  {  /* code omitted */  }
      public double getX()  {  /* code omitted */  }
      public double getY()  {  /* code omitted */  }
      public double distanceFromOrigin()  { /* code omitted */  }
}
```

Make a new data type, `FuzzyPoint`, that extends `Point`, but has the following slight changes.

- There should be a new member variable, fuzziness, a double. This represents how "Sharp" (like a sharp point of light from a laser) or "fuzzy" (blurry) the point is, like an out-of-focus spotlight.
- The constructor should now have three parameters: the x y coordinates, and the desired fuzziness.
- The distanceFromOrigin() method should not return the actual distance. Instead it should return the distance (as calculated using the formula above) plus or minus a small random amount depending on the fuzziness. In other words, if the true distance as calculated using the above formula is 35.0, and the fuzziness is 2.0, then the distanceFromOrigin method should return a random number between 33.0 and 37.0, rather than the true value 35.0.
- Hint: Don't copy-paste the formula, just call the parent's method for that part!

9) Make a new class, `HeisenPoint`, that extends `FuzzyPoint` (from question #8), and make the following changes.

- The constructor should have no parameters. Instead, the x and y coordinates should be initialized to random numbers between -100.0 and +100.0, and the fuzziness should be 5.0. This mimics the idea of a physics experiment where a particle is created by some reaction, but the position of the particle isn't controlled and instead a random location with some range.
- Override the getX() and getY() methods, so that rather than returning the actual x and y coordinates, they return a slightly randomized value instead. The returned value should be the actual value plus or minus a random number depending on the fuzziness (i.e. if the fuzziness is 5.0, then the range should be -5.0 to +5.0). This mimics a real particle in physics, where it is not possible to perfectly measure the actual position of a particle without some small amount random measurement error.
- In order to fully hide the actual, unrandomized x, y values from clients, is it necessary to also override the toString() and distanceFromOrigin() methods? Why or why not?

10) Using inheritance and the same `Point` data type from question #7, make a new data type `MovingPoint`, that is the same as `Point` but...

- There is one more variable, the "step size". This is meant to keep track of how fast (or how far) the point should move in each step.
- The constructor has three parameters instead of two: x, y, and the desired step size.
- There are four new methods: up(), left(), down(), and right(). These have no parameters. They modify the MovingPoint by changing the x, y coordinates, either adding (or subtracting) one "step size" amount to x or y, as appropriate.

11) Using inheritance and the same `MovingPoint` and `Point` data types from the previous questions, make a new data type `TrailingPoint`, that is the same as `MovingPoint` but designed such that it is not possible for a point to go backwards to the position it was last in. (It doesn't need to keep track of *all* previous positions, just the most *recent* previous position.)

- There are additional variables to keep track of the *previous* coordinates where the point was, before the last movement occurred.
- The constructor should still have three parameters, just like Moving Point.
- The movement methods, up(), left(), down(), and right(), should behave slightly differently than before. If the new position would put the point back where it was most recently, the point should not move at all. Otherwise, the point should move as expected. Hint: you might want to add a new helper method so you don't have to write nearly-identical code four times.

12) (a) Define a new class `Friend` that contains a name, and email, and a phone number. These three member variables are all be strings. Define a constructor so the following code works:

```
Friend bff = new Friend("Dennis", "ritchie@c.org", "867-5309");
```

b) Add member methods named `getEmail()` and `setEmail()`. The `getEmail()` method should take no parameters and it should return the friend's email address. The `setEmail()` should take one parameter, a String, and it should change the friend's email address to that string.

c) Suppose `contacts` is an array of `Friend` objects that has already been initialized. Write code that prints out all the names of friends in the array who have "@gmail.com" email address. If you need to add new member methods to the Friend class, then do so, but otherwise do not modify the Friend class. Hint: You can use any of the normal String class methods, such as `substring()`, `charAt()`, `endsWith()`, etc.

d) Write code that opens a file named "contacts.txt" then reads from it information about the user's friends. This information should be put into an array named contacts, which your code should declare and initialize. The file format is shown in an example at right. The first line is a count of how many friend's there are, and the remaining lines give the information for each friend, one per line.

```
6
King    lking@holycross.edu  (508)793-2248
Walsh   kwalsh@holycross.edu 508-793-3943
Royden  croyden@holycross    ext2472
Bill    bill@microsoft.com  unpublished
Larry   lpage@gmail.com   unknown
Sergey  brin@gmail.com    don't-know
```

13) a) What is the difference between public, protected, and private members of a class?

b) What's the difference between a final and a non-final member of a class?

c) What's the difference between a static and a non-static member of a class?

d) What's the difference between =, ==, and .equals() ?

14) The following code is an attempt to write a recursive function that will print a string in reverse. Unfortunately, it does not work. Explain what would happen if we use the function as it is written, then fix the function so it works properly.

```
public static void printStringReversed(String s) {
      char firstLetter = s.charAt(0);
      int n = s.length();
      String rest = s.substring(1, n);
      printStringReversed(rest);
      System.out.print(firstLetter);
}
```

15) Suppose you are given a function with this signature:

```
public static boolean contains(double[] arr, int s, int e, double t)
```

The function searches the array between positions s (inclusive) and e (exclusive), and returns true if the array contains value t somewhere in that portion of the array. For example, contains(x, 7, 10, 3.14) would search array x, but only positions 7, 8, and 9, to check if the value 3.14 is within that portion of the array. Write a new function with the following signature:

```
public static boolean containsDuplicates(double[] vals)
```

The function returns true if the *vals* array contains any duplicate values. Use the `contains()` function. Write this using a loop. Then try re-writing it using recursion instead; you may need to add a formal parameter. In both cases, the `contains()` function will be very handy.

16) Consider the following recursive function:

```
public static int Mystery( String  myString, char theChar,
                                    int first, int last) {      // line 1
    int answer = 0;                                             // line 2
    if (first >= last) {                                        // line 3
        return 0;                                               // line 4
    } else {                                                    // line 5
        answer = Mystery ( myString, theChar, first + 1, last); // line 6
        if ( myString.charAt(first) == theChar ) {              // line 7
            return answer + 1;                                  // line 8
        } else {                                                // line 9
            return answer;                                      // line 10
       }                                                        // line 11
    }                                                           // line 12
}                                                               // line 13
```

a) Which lines of code represent the solution to the base case? Also state what the base case is (i.e. under what conditions is the problem considered a base case?).

b) Indicate which lines of code represent the solution to the general case.

c) Suppose you have the following code use to call the Mystery function:

```
String theString = "CSCI";
int test;
test = Mystery(theString, 'C', 0, 4);
```

Draw a table showing a trace of the values of the following parameters as each recursive function call is executed:

```
first    last   myString.charAt(first)    answer    return value
```

d) What does the Mystery function do? Explain using simple and concise plain English.

17) Examine the following function and answer the questions below.

```
public static int calc(int angle, int incr) {
    StdOut.print("|");
    if (angle <= 180) {
        StdOut.print(angle + " " + incr);
        int x = calc(angle + incr, incr);
        StdOut.print("x is " + x);
        return 2 * x;
    } else {
        StdOut.println ("bounce");
        return 1;
    }
}
```

(a) What output is printed to the terminal during the following function call? What is the value of `response` after this line of code finishes executing? For partial credit, make a table or diagram to illustrate your work.

```
int response = calc(90, 30);
```

(b) Are there any integers for the parameters of this function that will cause infinite recursion? If so, give an example call. If not, explain why not.

Reinforce previous concepts material

18) Write a function named `countMatches()` that takes two String parameters, a and b. The function should compare the strings, character by character, and return a count how many of the pairs match between the two strings. For example, countMatches("abcdef", "aaffeb") would return 2, since there are two matching characters (the initial 'a' and the 'e' near the end). Similarly, countMatches("Happy", "Sappy") would return 4, since all the characters match except the first one. You should *not* assume the strings are the same length. For example, countMatches("Jakarta", "Java") should return 3 (because of the 'J' and two 'a' letters near the start), and the code should not crash just because one of the strings is longer than the other.

19) Write a function named `findDuplicates()` that takes one string as a parameter and returns nothing. The function should examine each adjacent pair of characters in the string. If the two characters are the same, the function should print a message indicating the character and the position in the string. For example, findDuplicates("Massachusetts") should print a message saying "ss appears at position 2" and also a second message saying "tt appears at position 10".

20) Write a function named printTitleCase() that takes a string parameter and prints that string with capitalization appropriate for a title. That is, the first letter of the string should be printed in uppercase. If a letter follows a space, then that letter should be printed in uppercase. Otherwise, each letter should be printed however it appears in the original string. As an example, printTitleCase ("FORTRAN programming tutorial"); would result in the output "FORTRAN Programming Tutorial". You can use Character.isUpperCase() and Character.toUpperCase() functions as needed. These functions have the following signatures:

```
public static boolean isUpperCase(char c); // returns true if c is upper case letter
public static char toUpperCase(char c); // returns upper-case version of letter c
```

21)

a) Write a snippet of code that asks the user to enter five words. Input these words and store them into an array. Then use a loop to count how many of the words in the array are equal to "duck".

b) Same as the previous question, but count how many of the words in the array start with the letter 'd'.

22) Write code to create a grid of floating point numbers, with 5 rows and 8 columns. Store the number 17.0 into every element of the array. Then print the array in a nicely formatted table, with spaces between the numbers and a newline after each row.

23) Suppose M is a 2D array of integers with 5 rows and 3 columns, and V is an array of three integers. Compute the matrix-vector product of M and V. The result should be a new array of five integers. The first number in the result array is the sum of multiplying each number in V with each of the corresponding numbers in the first row of M. The second number in the result array is the sum of multiplying each numbers in V with each element in the second row of M, and so on. For example:

M			V	result	note
2	3	4	1	42	`42 = 2*1 + 3*0 + 4*10`
2	7	5	0	52	`52 = 2*1 + 7*0 + 5*10`
6	9	8	10	86	`86 = 6*1 + 9*0 + 8*10`
2	4	9		92	`92 = 2*1 + 4*0 + 9*10`
8	0	1		18	`18 = 8*1 + 0*0 + 1*10`

24) The following function does not compile. Fix all the errors.

```
public static double average ( a, b ) {
     double avg = (a + b)/2;


 }
```

25) Write a function that counts how many positive numbers (greater than zero) there are in a partially-filled array of integers. The function should have two parameters, an array and a count of how many numbers are currently in the array. The function should not alter the list. It should return an integer. [variation: use a completely-filled array, and only a single parameter to the function; variation: count how many even numbers, rather than positives.]

26) Write a function that, given a partially-filled array of integers, returns a new completely-filled array that contains only the positive numbers from the original list. You can (and should) use the function from the previous exercise.

27) Write a function that counts how pixels in a given image have a particular color. The function should have two parameters, a reference to a Picture object (as defined by the Sedgewick & Wayne textbook), and a reference to a Color object (java.awt.Color). The function should search the entire picture for that color, and count how many times it appears. The function should return an integer. Remember: Don't use "==" when comparing Color objects... use ".equals(...)" instead.

28) Fill in the program below to simulate a gambler playing a game of chance. The program will simulate T trials of the game in all so that it can calculate statistics about the likely outcome of the game. The program uses three command-line arguments: an initial stake (e.g. $50), a goal amount (e.g. $250), and the number of trials T. During each trial, the gambler starts with the initial stake (e.g. $50), then she will make a series of fair $1 bets. If she eventually goes broke, she loses the game. If she reaches the goal amount (e.g. $250), then she wins. The gambler continues playing until either she wins or loses. Simulate T games, then print the number of times the gambler won and the average number of bets per trial.

Note: A *fair* $1 bet is one in which the gambler wins $1 half the time and loses $1 half the time.

Note: The expression Math.random() will pick a random integer between 0 and 1.0.

```
public class Gambler {
   public static void main(String[] args) {
          int stake = Integer.parseInt(args[0]); // Stating amount for each trial.
          int goal = Integer.parseInt(args[1]); // Desired amount.
          int T = Integer.parseInt(args[2]); // # of trials to perform
          int bets = 0; // total number of bets made over all trials
          int wins = 0; // total number of games won over all trials

                          // your code goes here....

          StdOut.println(wins + " wins out of " + T + " games");
          StdOut.println("Percent of games won = " + (100.0 * wins) / T);
          StdOut.println("Avg # bets per game = " + (1.0 * bets) / T);
   }
}
```

29) Given a 9-by-9 grid of integers between 1 and 9, write a code fragment to check if it is a valid solution to a Sudoku puzzle where each row and column contain the nine integers exactly once. (Okay, in real Sudoku, we have to check those 3x3 sub blocks, but we are going to ignore that for this problem and just stick with rows and columns). The only output should be a single message that declares the "solution correct" or there is an "error." You may assume a grid has already been declared:

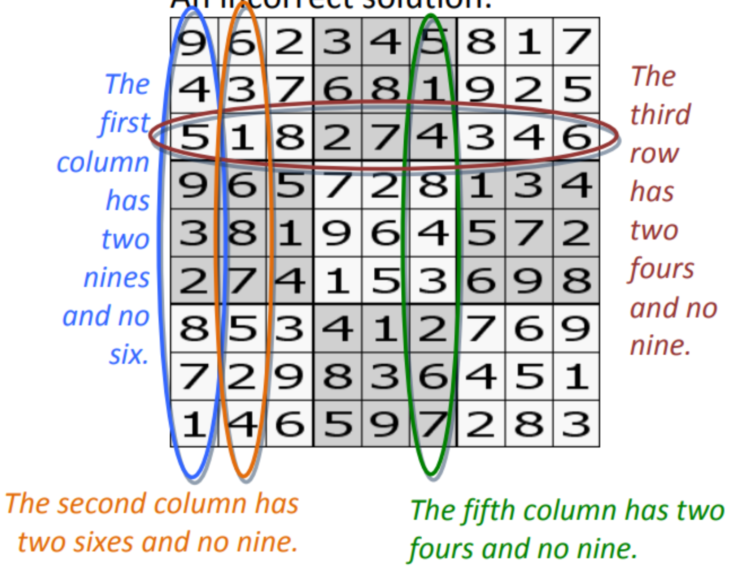
```
int [][] grid = new int [9][9];
```

You may also assume the grid has been filled with values.

A correct solution:

6	9	2	3	4	5	8	1	7
4	3	7	6	8	1	9	2	5
5	1	8	2	7	9	3	4	6
9	6	5	7	2	8	1	3	4
3	8	1	9	6	4	5	7	2
2	7	4	1	5	3	6	9	8
8	5	3	4	1	2	7	6	9
7	2	9	8	3	6	4	5	1
1	4	6	5	9	7	2	8	3

An incorrect solution:


The first column has two nines and no six.
The third row has two fours and no nine.
The second column has two sixes and no nine.
The fifth column has two fours and no nine.

9	6	2	3	4	5	8	1	7
4	3	7	6	8	1	9	2	5
5	1	8	2	7	4	3	4	6
9	6	5	7	2	8	1	3	4
3	8	1	9	6	4	5	7	2
2	7	4	1	5	3	6	9	8
8	5	3	4	1	2	7	6	9
7	2	9	8	3	6	4	5	1
1	4	6	5	9	7	2	8	3

30) Given a file containing just a list of words (i.e. no unusual punctuation), write a function to count how of those words begin with a certain letter. Do not differentiate between upper and lower case letters when counting. Your function should be named countStart() and it should have two parameters: a String, which is the name of the file to search, and a target character.
Your function should return the number of words in the file that begin with that target character.
You do not need to do any error checking.

```
help
batman
Candyman
manner
maneater
beef
bigman
summer
month
May
mother
Montana
Spring
```

For example, for the file at right (which is named "words.txt"), then

countStart("words.txt", 'M') should return 6
countStart("words.txt", 'b') should return 3
countStart("words.txt", 'S') should return 2
countStart("words.txt", 's') should return 2 as well

These functions in the Character library will be useful:

char toUpperCase(char c);
char toLowerCase(char c);

31) After we run the program containing the following snippet of code

```
int val = 5;
Out f = new Out ("temp.txt");
StdOut.println(val);
f.close();
```

The file "temp.txt" is empty. Why?

32) Write a function to find the largest number in an array. The function will return the index of the position that contains the largest number. The function should be named `most()` and it should have one parameter, *a list of integers*. You do not need to do any error checking.

For example,

If the parameter is an array that holds five numbers:

5	6	22	8	10

Function `most()` *should return:* 2

33) Write a function named `histogram()` to count the number of times each letter occurs in a file and return an array (of size 26) with the frequency count for each letter. The name of the file is passed as the parameter to the function. Count all alphabetic characters, but do not differentiate between upper and lower case letters when counting letter occurrences, i.e. if `'f'` occurs 5 times in the file and `'F'` occurs 3 times, then the frequency count corresponding to the letter `'F'` should be an 8. Characters in Java are represented by numerical ASCII values. The ASCII values for the upper case letters are contiguous, so that `'C'-'A'` yields 2 because `'C'` comes 2 characters after 'A' in the alphabet, likewise `'R'-'A'` yields 17 because `'R'` comes 17 characters after 'A' in the alphabet. The signature for the function will be:

```
public static int[] histogram (String name)
```

```
d Rr55DD J c
kk 098.-z
Hello
```

Given the contents of the example file on the left, your function should return an array that contains the following values:

numerical ASCII value	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
corresponding letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
array position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	0	0	1	3	1	0	0	1	0	1	2	2	0	0	1	0	0	2	0	0	0	0	0	0	0	1

In other words, array position 17 contains value 2, because there were two 'R's (counting both upper and lower case) in the file. In summary, there are 0 in all elements except the count for entries for C, E, H, J, O, and Z which will be 1, the entries for K, L, and R which will be 2, and the entry for D which will be 3.

34) In project three you wrote a program to decode a Huffman encoded file. For this problem, you will write a program to count the frequency of letters in a file that is necessary prior to creating the Huffman encoding itself. The name of the file containing the message is a command line argument to the program. The frequency information will be put in an output file named Huff.txt. Your program should:

(1) Create the output file to hold the results.

(2) Count the number of times each letter of the alphabet occurs in the input file. Only keep track of information for 'A' through 'Z', and count 'a' and 'A' the same, ignoring upper-case/lower-case distinctions. Store the frequency counts in an array.

(3) Write each letter and its corresponding count, in descending order, into the output file. This means that the letter that has the highest count should be output first, then the letter with the second highest count next, and so on. Do this until for all letters with a frequency count above zero. That is, ignore any letters with a zero frequency count.

(4) At the end of the program, display a message with all the letters that did not appear in the file (that is, all letters with frequency count zero).

Notes:

- You can (and should) use the following functions from earlier questions on this exam:
```
// Searches a list for the largest number, and returns the index of that number within the list.
// Example: if list contains { 3, 5, 2, 99, -1, -1, 7}, then the function would return 3.
public static int most(int[] list);
// Create a letter-frequency histogram for the given file. The returned array will have length 26,
// holding 26 values, one for each letter of the alphabet. Each value in the array is the count of
//how often the corresponding letter appeared in the file. So position 0 is the count of the 'a'
// and 'A' characters, position 1 is the count of the 'b' and 'B' characters, and so on.
public static int[] histogram(String filename);
```
- There is no need to do any error checking. The input file will exist, and won't be empty, and will contain at least some valid letters.
- See the appendix for useful methods in the Math, String and Character libraries.
- Recall that arithmetic works for characters. For example, (char)(3 + 'A') evaluates to the character 'D'. Similarly, 'E'-'A' evaluates to 4, since 'E' comes 4 places after 'A' in the alphabet.

Example:

If the input file contains this text:

```
d rr55DD Jc
kk  098.-z
Hello
```

Then the program should print to Huff.txt:

```
The letters from most frequently used to least frequently used.
D appeared 3 times.
K appeared 2 times.
L appeared 2 times.
R appeared 2 times.
C appeared 1 times.
E appeared 1 times.
H appeared 1 times.
J appeared 1 times.
O appeared 1 times.
Z appeared 1 times.
These letters did not appear at all:
A B F G I M N P Q S T U V W X Y
```

35) Write a program named TimeKeeper to help students with time management by finding the best order in which to complete their homework, practice, and other tasks. Each task has a name (e.g. "Math" or "Lunch"), and a deadline (an integer between 8 and 18, representing the time of day from 8AM to 6PM). Each task takes exactly 1 hour to complete, so it must be scheduled to begin at least one hour before its deadline otherwise there would not be time to finish the task. Begin your day at 7AM.

The program should accept the list of task names and deadlines on the command line. (Hint: args.length will be useful here.) It should then calculate and print out an optimal order for tasks to be completed. For each hour of the day (starting at 7AM), pick a task using the following strategy: from among all tasks that have not yet been scheduled and whose deadline has not yet arrived, pick one with the soonest deadline, and schedule that task for the current hour. For example, for 10AM, the program should look at all tasks with deadline of 11AM or later that have not yet been scheduled, and pick one with as early a deadline as possible. Whichever task is chosen is the one the student will start doing at 10AM.

There may be tasks that cannot be completed before their deadlines. Any task that cannot be completed before its deadline is considered failed. The program should stop after scheduling as many tasks as possible, printing a final summary showing: a list of tasks that failed; and a summary of how many tasks were able to fit within the day and how many tasks failed. Here is an example of the program in action. (User input is shown in blue italics.)

```
java TimeKeeper Bio CS Lunch Music Math Lit Dance 11 9 14 8 9 8 13
At time 7, you should do Music (deadline 8)
At time 8, you should do CS (deadline 9)
At time 9, you should do Bio (deadline 11)
At time 10, you should do Dance (deadline 13)
At time 11, you should do Lunch (deadline 14)
Not enough time for Math (deadline 9)
Not enough time for Lit (deadline 8)
You will finish 5 tasks and fail 2 of them.
```

36) This problem involves encoding words in Pig Latin, a fun secret language for us all. Rather than tackling the whole encoder at once, we will write it in three parts.

(a) Write a function named **`firstVowel()`** that finds the position of the first vowel in a word. (The vowels are 'a', 'e', 'i', 'o', and 'u'.) The function should take one parameter, a *String*, and return an *integer* representing the position of the first vowel. If the String does not contain a vowel, return -1. Examples:

Example: if the String is "`under`" return `0` .
Example: if the String is "`strand`" return 3 .
Example: if the String is "`stdnvb`" return `-1` .

(b) Write a function named **`pigLatin()`** to convert a word using the following criteria:

- If it begins with a vowel, append "-hay" to the end.
  Examples: "under" becomes "under-hay"; "apple" becomes "apple-hay".
- If it begins with a sequence of consonants, followed by a vowel, append a dash, move the consonants to the end, then append "ay" to the end.
  Examples: "standard" becomes "andard-stay"; "chub" becomes "ub-chay"
- If the word does not have a vowel, then it cannot be converted to Pig Latin.
  Example: "sly" can't be converted to pig latin.

The function should take one parameter, a *String* and print out both the original word and its Pig Latin equivalent. If the word can't be converted, print a message saying so.

Examples.

The call **`pigLatin("standard")`** would print the following to the console:

```
standard in pig-latin is andard-stay
```

The call **`pigLatin("output")`** would print the following to the console:

```
output in pig-latin is output-hay
```

The call **`pigLatin("crypt")`** would print the following to the console:

```
crypt cannot be converted to Pig Latin
```

(c) Write the main function for a program that reads in a sequence of *Strings* from a file named **`words.txt`** and prints each word and its Pig Latin translation to standard output. In other words, for each word in the file, call the function from part (b) to print the word and the pig latin form of the word. For example, if `words.txt` contains:

```
biology  stranded  ouji  hhhhh
```

then your program should print to the console:

```
biology in pig-latin is iology-bay
stranded in pig-latin is anded-stray
ouji in pig-latin is ouji-hay
hhhh cannot be converted to Pig Latin
```

37. What is the output from the program with several calls to a recursive function?

```
 public class Mystery
   public static int mystery(int a, int b)
      if (b==0)
         return 0;
      if (b % 2 == 0)
         return mystery (a+a, b/2);
      return mystery (a+a, b/2) + a;
   }
   public static void main(String[] args) {
      StdOut.println(mystery(3, 2));
      StdOut.println(mystery(2, 25));
      StdOut.println(mystery(3, 11));
   }
}
```

38 Suppose the string variable `login` contains a student's login name in typical Holy Cross format (e.g. "jqsmit17"). Write a snippet of code that assigns the appropriate values to `the first, middle, and last` variables. For example, if `login` is "jqsmit17", then the code should assign first to be the letter 'j', it should assign middle to be the letter 'q', and it should assign last to be the word "smit". You can assume the login has exactly the format shown in this example (2 initials, 4 letters, then 2 digits).

```
String login;
char first, middle;
String last;

login = StdIn.readString();
// assigns login to be something like "jqsmit17"

// your code here
```

Try the Exercises from topic 19: Make a function that...

... **duplicates** a given string n times, for any positive n
e.g. "dub" becomes "dubdubdub".

... returns the ***first half*** of a string, e.g. "Cattle" becomes "Cat"

... finds the ***index*** of a the first occurrence of a given letter within a given string (this is what String's indexOf() function does).

... ***splits*** a string at the first comma and returns first part, before the comma
(hint: use substring and indexOf)

... returns the ***reverse*** of a given string

... returns the ***frequency*** of a given letter within a given string

Also from topic 19:

A *genome* is a long sequence of A, C, T, and G.
A *gene* is a portion of a genome...
that is proceeded by ATG (the *start codon*)
whose length is a multiple of 3
that is followed by TAG, TAA, or TGA (the *stop codons*)
Goal: Given a file containing a genome, find and print out all the genes.
Algorithm: Start with beg = -1. Work left-to-right. If start codon is found, set beg to that index. If stop codon is found and length is multiple of 3, print the gene and reset beg to -1.