FINAL EXAMINATION – DECEMBER 2006
CSC 112
NICHOLAS R. HOWE

This is a closed-book exam. You may use two double-sided 8.5x11 sheets of notes.

All answers to this exam should be written in your exam booklet(s). Start with the questions that you know how to do, and try not to spend too long on any one question. Partial credit will be granted where appropriate. You will have two hours and twenty minutes. Good luck!

Data Structures (8 points)

1.) We have used a number of data structures from the Java collections framework, including `LinkedList`, `ArrayList`, and `HashMap`. Of these three, indicate which would be most appropriate for the following uses, and why. If two seem equally appropriate, then explain why as well.

a.) We need to buffer requests to a particular service. The requests should be handled in the order they are made. Although many requests will be handled over time, the number waiting at any one instant is expected to be small.

ArrayList or LinkedList. HashMap is not good because it doesn't keep order. LinkedList gives the simplest implementation, but requires more memory allocations and garbage collection. An ArrayList offers the most efficient data structure if it is used in such a way that elements are never copied.

b.) We are implementing a password-protected database. Since every access to the database will include a user account and a password that must be checked, and we are expecting many people to use our system, we require a data structure that will let us quickly match an account to the right password.

HashMap, because it is fastest.

c.) We are simulating genetic drift on a piece of DNA. Our algorithm scans over a sequence of base pairs (which may be quite long), randomly inserting mutations as it goes. A mutation may consist of eliminating a base bair, replacing it with a different base pair, or inserting a base pair.

LinkedList, because it handles insertion/deletion best.

d.) We are implementing a digital jukebox, and need a data structure to keep track of our song data. Users of our software will have the option of playing a song specified by number, or of playing all songs in sequence.

ArrayList, because it can do both sequential and random access efficiently.

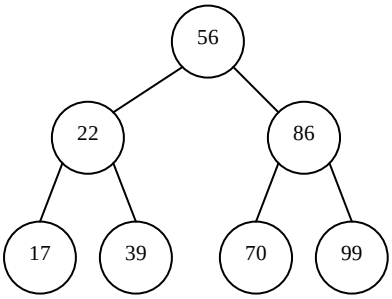
Trees (12 points)

2.) The following questions review several concepts related to trees.

a.) Create a balanced binary search tree from the following array:

17	22	39	56	70	86	99

b.) Print the elements of the tree you just drew in **preorder**.
56, 22, 17, 39, 86, 70, 99



c.) Compute the numeric value of the postfix expression:
99.0 3.0 3.0 2.0 + 8.0 2.0 - * + / = *3.0*

Heaps (10 points)

3.) Demonstrate the heap sort algorithm on the array below. You should show the contents of the array after each swap performed. (Assume that the heap is grown from left to right, with the root at 0, and the sorted portion is grown from right to left.) Indicate the configuration at the midpoint of the algorithm, when the entire array has been heapified. (Hint: it may be helpful to draw the heap as a tree, before writing your answer down as an array.)

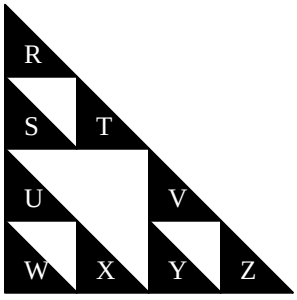
3	8	6	1	4

8 3 6 1 4 // 8 4 6 1 3 (mid) // 3 4 6 1 8 // 6 4 3 1 8 // 1 4 3 6 8 // 4 1 3 6 8 // 3 1 4 6 8 // 1 3 4 6 8

Recursion (12 points)

4.) The following code, when called with rank = 2 and appropriate values for r and g, will produce the image at right (minus the alphabetic labels).

```
private void sierpinskiTriangle(int rank, Rectangle r, Graphics g) {
   if (rank <= 0) {
       Polygon p = new Polygon();
       p.addPoint(r.x,r.y);
       p.addPoint(r.x,r.y+r.height);
       p.addPoint(r.x+r.width,r.y+r.height);
       g.fillPolygon(p);
   } else {
       int hw = r.width/2;
       int hh = r.height/2;
       Rectangle r1 = new Rectangle(r.x,r.y,hw,hh);
       Rectangle r2 = new Rectangle(r.x,r.y+hh,hw,hh);
       Rectangle r3 = new Rectangle(r.x+hw,r.y+hh,hw,hh);
       sierpinskiTriangle(rank-1,r1,g);  // line A
       sierpinskiTriangle(rank-1,r2,g);  // line B
       sierpinskiTriangle(rank-1,r3,g);  // line C
   }
}
```



a.) Suppose that line B is removed from the program. What portion of the image will still be drawn? (Indicate by letter.) *R, T, V, Z.*

b.) If lines A and C are both removed, what portion will be drawn? *W*

c.) What would be the effect of reversing the order of lines A, B, and C? *The triangles would be drawn in a different order, but the end result would be the same. Since the window is not normally updated until all drawing si complete, no difference would be visible.*

Programming Style (12 points)

5.) Programming languages contain many features designed to eliminate the need for redundant code (i.e., code that is essentially the same except for minor differences). Give three examples of such mechanisms in Java, explaining exactly how they help avoid redundant code. Give three advantages that result from using such techniques.

Loops, methods, and generic types all eliminate certain forms of redundant code. Loops shorten code by allowing repeated sections to be written only once. Methods shorten code by allowing a section of the same code to be called by name as desired, with optional paramaeters to vary behavior. Generic types allow a class structure to be reused with different underlying data. Each of these techniques shortens programs, ensures consistency, and makes maintenance and debugging easier.

Hash Tables (10 points)

6.) Suppose that you create a hash table with five entries, and use **k mod 5** as your hash function. Your table will use open adressing with linear probing. Draw the state of the table after **each** of the following operations, assuming it begins empty:

a.) Insert **Bermuda** under key 441.

b.) Insert M**ontserrat** under key 664.

c.) Insert **Jamaica** under key 809.

d.) Insert **Jamaica** under key 876.

e.) Insert **Dominican Republic** under key 809.

key	data
441	Bermuda

key	data
441	Bermuda
664	Montserrat

key	data
809	Jamaica
441	Bermuda
664	Montserrat

key	data
809	Jamaica
441	Bermuda
876	Jamaica
664	Montserrat

key	data
809	Dominican Rep.
441	Bermuda
876	Jamaica
664	Montserrat

Programming Style (14 points)

7.) For each of the following situations, decide whether the datum described would be best implemented as a local variable, an argument to a method, a class field, or a static class field.

a.) In a GUI, each object present in the window has an associated position that must be stored. *Class field.*

b.) In the constructor for a class, a loop will step through the elements of an array to process them. The loop needs a counter to keep track of the current index. *Local variable.*

c.) A ticket-issuing program must keep track of how many tickets have already been issued, so that each time a `Ticket` object is constructed it can be given a unique serial number. *Static class field.*

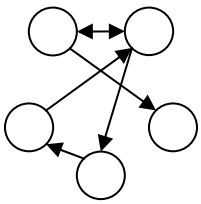
d.) A recursive function must keep track of the number of times it has been called, so that it can return when this number exceeds a predefined threshold. *Argument to method.*

e.) A program needs to create a `BufferedReader` to read from the standard input. All the reading will take place within `main()`. *Local variable.*

f.) A program needs to create a `BufferedReader` to read from a file. Depending upon factors that are unpredictable when the program is written, various different objects may need to read input from the file at various times. *Static class field.*

g.) A program will read a data structure from a `BufferedReader`. Depending on the circumstances, the `BufferedReader` may get its data either from the standard input or from a file. *Argument to method.*

Graphs (12 points)



8.) These two questions deal with graph representation.

a.) Draw the graph represented by the following adjacency matrix:

0	1	1	0	0
1	0	0	1	0
0	0	0	0	0
0	0	0	0	1
0	1	0	0	0

b.) Convert this to an edge-primary representation.

-1	1	0	0	0
-1	0	1	0	0
1	-1	0	0	0
0	-1	0	1	0
0	0	0	-1	1
0	1	0	0	-1

Exceptions (10 points)

9.) Write one or two paragraphs descibing the purpose of exceptions in programming languages. Under what circumstances should they be used, and when should they be avoided?

Exceptions are used to deal with unforeseen, out-of-the-ordinary circumstances. For example, they are often used when a data structure is found to have an illegal value, when it is used in an improper manner, or when some other unusual circumstance occurs that precludes the normal continuation of execution. They allow a program to return control to an error handler with the proper context to deal with the exceptional condition. By isolating error-handling code away from the code for normal execution, exceptions can make programs neater and easier to read. Programmers should avoid overusing exceptions, especially in non-exceptional situations, since this can actually make the behavior of a program more difficult to understand, and less efficient.