CMPS 13H - Fall 2002
Final Exam
December 4, 2002

Name:_____ID:_____

This is a closed note, closed book exam. Any place where you are asked to write code, you must declare all variables that you use. However, I just want code fragments, you must not write extra code such as a class specification or extraneous print statements.

Section I: 40 multiple choice questions [2 points each]

For each of the following questions, write the letter of the best answer to the left of the question including, if appropriate, "All" for "All of the above" or "None" for "None of the above".

1. Comments are useful in a program to:
   a. Help the java compiler figure out what you are doing
   b. Help someone else figure out what you are doing
   c. Help you figure out what you are doing

2. Which are all keywords:
   a. `float, catch, static, while, extrudes`
   b. `final, default, main, null, continue`
   c. `for, throws, case, break, public`

3. Which are all invalid identifiers:
   a. `true, Testing_1$_$, don'tknow`
   b. `This-is-ok, 1two, #iffalse`
   c. `hello$, 123, foo%`

4. Which can store the largest number:
   a. `float`
   b. `double`
   c. `int`

5. Which is sorted in decreasing order of precedence:
   a. `+ / %`
   b. `+ * /`
   c. `++ * =`

6. Given `a=3, b=5, c=7`, which boolean expression evaluates to true:
   a. `(a >= b) || !(a <= b)`
   b. `!(a == b) || (b > c)`
   c. `!((a != b) || (b < c))`

7. The boolean expression in an `if` statement is evaluated:
   a. Each time through the loop
   b. Only once
   c. Only when the statement is true

8. In a single pass through an `if-else` statement
   a. It is possible that neither statement will be executed
   b. One of the statements must be executed
   c. Both statements may be executed

9. In a `while` statement:
   a. The boolean expression is always executed at least once
   b. The statement is always executed the same number of times as the boolean expression
   c. The statement is always executed at least once

10. `While` statements are useful when you
   a. Know how many times the loop will executed
   b. Don't know how many times the loop will execute
   c. Don't know if you want to execute a statement or not

11. In a `for` loop:
   a. The update expression is executed every time the boolean expression is evaluated
   b. The statement is executed as many times as the boolean expression is true
   c. The initilization expression is executed each time at the top of the loop

12. `Break` statements:
   a. Cause the program to exit a loop it is currently executing
   b. Cause the program to exit the method it is currently executing
   c. Cause the program to exit

13. `Continue` statements:
   a. Cause the program to jump to a designated spot in the program
   b. Cause the program to exit a loop it is currently executing
   c. Cause the program to jump to the top of a loop it is currently executing

14. `Switch` statements:
   a. Aren't particularly useful with strings
   b. Can have a string in the controlling expression
   c. Can have a string in the `cases`

15. The default case in a `switch` statement:
    a. Says what to do if the controlling expression is false
    b. Says what to do if the controlling expression doesn't match any of the cases
    c. Says what to do if the `switch` statement fails

16. Static methods:
    a. May not access non-static member variables
    b. Are the same as constants
    c. Always return the same value

17. Which is true about variable scope:
    a. Any variable declared in the initialization expression of a `for` loop is not accessible anywhere outside that `for` loop
    b. Any variable declared in a class is accessible by all of the methods in that class
    c. Any variable declared in a method is accessible anywhere in that method

18. Parameters are:
    a. References to the variables specified in the method call
    b. Copies of the variables specified in the method call
    c. Copies of the objects referred to by the variables specified in the method call

19. Recursion is when:
    a. A method calls itself
    b. More than one method call the same method
    c. More than one method have the same name

20. Which is false about arrays
    a. When you pass an array as a parameter to a method, the method cannot change the values in the array
    b. Once created, the size of an array cannot change
    c. The type of objects in an array must all be the same

21. Given `int[][] foo = new int[3][5];` which is true
    a. `foo[3][5]` is the last element of the array
    b. `foo[3]` has five elements
    c. `foo` is a 2D array with 8 elements

22. Given `String A = "Hello", B = "Hello";` what does `A==B` equal
    a. `"Hello"`
    b. `true`
    c. `false`

23. Private member variables are:
    a. Accessible by any method in the same package
    b. Accessible only by private methods in the same class
    c. Accessible only by private methods in the same class

24. What is the minimum running time of an insert into a linked list:
    a. O(1)
    b. O(log$n$)
    c. O(n)

25. What is the maximum running time of an insert into a linked list:
    a. O(1)
    b. O(log$n$)
    c. O(n)

26. What is the main difference between linked lists and arrays
    a. Linked lists are more space efficient, but more flexible
    b. Linked lists are bigger, but faster
    c. Linked lists are more flexible, but less space efficient

27. Stack and Queues are just linked lists:
    a. With no difference
    b. But slightly more efficient
    c. With constraints on the operations

28. Stacks are useful for:
    a. Storing information in sorted order
    b. Implementing method calls and returns in a programming language
    c. Storing events (such as keys typed on the keyboard) in the order in which they were received

29. Which is true:
    a. An O(log$n$) algorithm is always simpler than an O(n) algorithm
    b. An O(log$n$) algorithm is always faster than an O(n) algorithm
    c. An O(log$n$) algorithm is not always faster than an O(n) algorithm

30. Which is false:
    a. Bubble Sort can be faster than Merge Sort, depending on the input
    b. Selection Sort is in general slower than Quicksort
    c. Quicksort is always faster than everything but Radix Sort

31. What is the worst case running time of Quicksort:
    a. O(n)
    b. O($n$log$n$)
    c. O(n^2)

32. What is the best case running time of Bubble Sort
    a. O(n)
    b. O($n$log$n$)
    c. O(n^2)

33. What is the best case running time of Insertion Sort
    a. $O(n)$
    b. $O(n\log n)$
    c. $O(n^2)$

34. What is the worst-case running time of Merge Sort
    a. $O(n)$
    b. $O(n\log n)$
    c. $O(n^2)$

35. What is the expected running time of Radix Sort on n d-bit integers as a function of n and d
    a. $O(n/d)$
    b. $O(nd)$
    c. $O(n\log d)$

36. Which is true about Binary Search:
    a. It is faster on binary search trees than arrays
    b. It is faster on arrays than linked lists
    c. It is faster on linked lists than arrays

37. The two main advantages of Trees over Arrays and Linked Lists are:
    a. Efficient use of memory and efficient searching and sorting
    b. Flexible size and efficient use of memory
    c. Flexible size and efficient searching and sorting

38. In a binary search tree, all nodes in the left subtree of a given node have:
    a. Values greater than the node
    b. Values that may be greater or smaller than the node
    c. Values smaller than the node

39. The main advantage of 2-3, 2-3-4, and Red-Black trees and binary search trees is:
    a. That they can be searched in $O(n)$ time
    b. Good worst-case running time of the various operations
    c. Efficient use of storage space

40. Chain-Bucket Hashing
    a. Increases the performance of a hash table
    b. Increases the number of hash table locations that must be probed when a collision occurs
    c. Increases the flexibility of a hash table

Section II: 6 questions [20 points each] (You may select one of these to be extra credit)

1. Writing java code

Write a method called primes that takes as a parameter an integer and prints out all primes between 1 and that number. Remember that a prime is a number that is evenly divisible by 1 and itself. Your method does not have to be efficient, it just has to work.

2. Efficiency of various data structures

Fill in this table with the average running times of the specified operations on the specified data structures

	Insert	Delete	Retrieve	Traverse
Unsorted Array				
Unsorted Linked List				
Sorted Array				
Sorted Linked List				
Binary Search Tree				
2-3 tree				
Hash Table				

3. Linked Lists

Given a linked list implementation as follows (lots of details omitted):

```
public class LinkedList {
    private Node head;
    private int count;

    public LinkedList();
    public void insert(int index, Object item);
    public void delete(int index);
    public Object get(int index);
    private Node find(int index);
    public void removeAll();
    public int length();
}

public class Node {
    private Object item;
    private Node next;

    public Node(Object newItem);
    public Node(Object newItem, Node nextNode);
    public void setItem(Object newItem);
    public Object getItem();
    public void setNext(Node nextNode);
    public Node getNext();
}
```

Write a method called reverse() that reverses the order of the items in the list, without allocating any new nodes.

4. Recursion and Trees

Given the following Binary Tree Code (lots of details omitted):

```
public class BinaryTree {
    protected TreeNode root;

    public BinaryTree();
    public boolean isEmpty();
    public Object getRootItem();
}

public class TreeNode {
    private Object item;
    private TreeNode leftChild;
    private TreeNode rightChild;

    public TreeNode(Object newItem)
    public TreeNode getLeft()
    public TreeNode getRight()
}
```

Write a recursive method called treeheight that computes the height of the tree

5. Trinary trees

I have a great idea! If Balanced Binary Search Trees are good, then Balanced Trinary Search Trees must be even better! The idea is that each node will contain one or two values and have 0 to 3 children (like three-nodes in 2-3 trees). Every node except leaf nodes will always contain two values and we'll make sure that insert and delete always keep the trees balanced such that the height of all three subtrees of a node never differ by more than 1.

Are Balanced Trinary Search Trees better, worse, or no different from Balanced Binary Search Trees? Why? In answering this question, discuss at least the storage space consumed by the tree and the minimum, maximum, and average running time of a search on a Balanced Trinary Search Tree.

6. Mystery Data Structure

I am working on a simulation of a real-time scheduler. A scheduler is a component of an operating system that chooses which “process” or program to run next.

In this simulation, new processes arrive at more or less random times, and each process has a *deadline* by which it must finish running.

My simulation uses Earliest Deadline First scheduling – it always chooses to run the process with the earliest deadline.

What data structure should I use to manage the processes in this simulation? Why? Explain your answer.

7. 2-3 trees

Given an intially empty 2-3 tree, show what tree would result after you

a) Insert 3

b) Then insert 43

c) Then insert 1

d) Then insert 88

e) Then insert 54

f) Then delete 1