CS211 Prelim 2. 7:30-9:00PM, Tuesday, 20 April 2004. Uris Auditorium.

The only reason for not taking it then is that you have a conflict with another evening prelim. If this is the case, please let Gries know by Thursday, 15 April.

See the end of this document for sample questions.

You are responsible for everything that was on the first prelim. We aren't looking to test that material, but it is understood that you know all the technical Java concepts that we have covered. That material should be part of your programming repertoire by now.

Below, with each section, we list what you are responsible for. At the end, we give some sample questions.

Questions on searching and sorting.

You should know the following algorithms: linear search, binary search, partition, selection sort, insertion sort, merge sort, quick sort, and heap sort. This means that *you must be able to produce our versions of them*, complete with invariants if they have loops.

There are two reasons for asking you to be able to *develop* these algorithms. (1) These are basic algorithms with which every programmer should be familiar. (2) If you can develop these algorithms from their specs whenever you are asked for them, you are well on your way to developing good programming habits. We do not memorize the code. We memorize the specifications, perhaps one or two keys points, and are then able to develop the algorithm. That is what we would like you to be able to do.

The way to learn this is to practice. Don't just study by reading. Get a fresh piece of paper and try to develop the algorithms yourself. Note that we may give you a variation of the original problem. For example, we may ask you to write selection sort to sort b, to sort b[0..k-1], or to sort b[h..k]. The idea is the same in all of them; the details are different, and it does not help to memorize code.

Linear search, binary search, and partition are on slides 12-28 for lecture 06. Selection sort and insertion sort are given in these notes. Merge sort and quicksort are given in the handout for recitation 8. Heap sort is in the handout for lecture 20.

You should know the worst-case and best-case execution times of these algorithms.

Questions on algorithmic analysis.
You are responsible for: Weiss, chapter 5, as follows:
•5.1 What is algorithmic analysis?
•5.2 Examples of running time
•5.3 NO, not responsible for this NO.
•5.4 Definition of Big-oh and Big-theta. You should be able to use these definitions.
•5.5 Everything except harmonic numbers. This includes the repeated doubling and repeated having stuff.
•5.6. NO, not responsible for this section. But be able to determine the order of execution time of: linear search, binary search, partition, insertion sort, selection sort, merge sort, quick sort (see the handout for recitation 8).
•5.7 Checking an algorithm analysis.
•5.8 Limitations of big-oh analysis.

You should be able to prove that a given function is O(f(n)) (for some f(n)), and you should be able to analyze an algorithm and figure out its order of execution time. *We are not looking for anything tricky here, just basic understanding*.

Questions on data structures.

1. Linked lists. We have used the terms linked list, linked list with header, doubly linked list, and doubly linked list with head and tail with technical meanings. You have to know what those terms mean. You should be able to write algorithms that traverse these kinds of lists, dooin something to each element. You have to be able to write code to remove an element from and add an element to these kinds of linked lists. Don't memorize code; instead, be able to draw the situation and develop the code from the drawing. Know the advantages and disadvantages of using these kinds of linked lists.

Be able to say what the order of execution time is for these operations on the different kinds of linked list: find a value, delete the minimum value, delete a node whose name is known, insert a value after a given node, insert a value before a given node. All of these are straightforward and obvious if you understand the details of the different kinds of list.

2. Hash tables. Know how to implement a set in a hash table, as discussed in the handout for recitation 6. You do not have to know a hash function, but you have to know what its purpose is. Know what linear probing and quadratic probing mean. Be able to show how to add or delete an element. Know what the load factor is and how many probes can be expected when the load factor is 1/2.

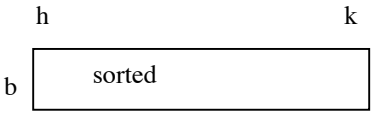
3. Binary trees. Weiss, sections 18.1.1 (not 18.12 and 18.13), 18.2, 18.3). And the slides from lecture 18.

Know the definition of a binary tree and how a binary tree is implemented. Be able to write methods that perform preorder, inorder, and postorder traversals of trees. (Don't read Weiss, section 18.4, to find out about preorder, inorder, and postorder traversal, because he doesn't do them recursively.) Know how to implement an expression in a binary tree.

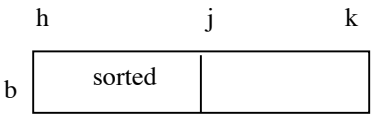
4. Binary search trees. Weiss, section 19.1 and slides from lecture 19. Know the definition of a binary search tree. Know how to look for a value in a binary search tree, to find the minimum value of a binary search tree, and to add a value to a binary search tree.

5. Priority queues and heapsort. Know what is in the slides for lecture 20. You are responsible for knowing heap sort, as given in those slides.

Insertion sort
Precondition: h–1 <= k.
Permute the values of b[h..k] so that:
Postcondition R: b[h..k] is sorted:

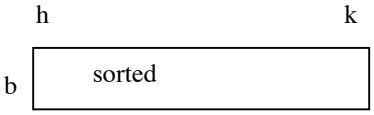


Here is the invariant P:

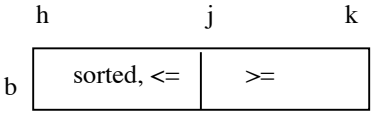



```
for (int j= h; j <= k; j= j+1) {
    Push b[j] down into its sorted
        position in b[h..j];
}
```

Selection sort
Precondition: h–1 <= k.
Permute the values of b[h..k] so that:
Postcondition R: b[h..k-1] is sorted:



Here is the invariant P:




```
for (int j= h; j <= k; j= j+1) {
    Store in m the index of the minimum of b[j..k];
    Swap b[j] and b[m]
}
```

Questions on searching and sorting.

Q1. The departmental phone number is 2557316. Show how the order of the digits changes as the digits are sorted using Quick Sort, Insertion Sort, and Merge Sort. For Quick Sort, assume that the first item is used as the pivot item (this is not a good choice).

Q2. For each of the following problems, write the invariant (as we have given it) and then write the loop based on that invariant and bound function:
(a) linear search
(b) binary search
(c) partition algorithm
(d) selection sort

Q3. Write mergesort and quicksort.

Q4. Consider the following sorting methods: Insertion Sort, Merge Sort, and Quick Sort.
(a) What is the running time for each method when all the the array values are equal?
(b) When the values are in order?
(c) When the values are in reverse order?

Questions on algorithmic analysis.

Q1. (a) Java Nagila, a CS211 student, implemented a bunch of algorithms and came up with the following running times as a function of input size. Using big-O notation, write down the simplest and most accurate expressions for the complexity of these algorithms.

$f(n) = 2n \log(n) + 4n + 17 * \log(n)$
$g(n) = \log(n*n*n)$

(b) You have an algorithm that runs in time O(log n). Java Nagila claims to have found a better algorithm that runs in time O(log(n-1)). Is either algorithm better than the other in an asymptotic sense? Explain briefly using the formal definition of big-O notation.

(c) The following method returns element number i of an integer list (the list is not empty).

```
public static int element (List list, int i) {
    if (i == 1) return list.first();
    return element(list.rest(), i-1);
}
```

What is the asymptotic time complexity of accessing an element m of a list using this method? Express the complexity as a function of m and n, the list length.

Q2. Fill in the table below with the *expected* time for each operation. Use big-O notation. The operations are insert (place a new item in the data structure), find (test if a given item is in the data structure), getMin (return the value of the minimum item in the data structure and delete it from the data structure), and successor (given an item, return the successor of that item).

	Insert	**find**	**getMin**	**successor**
sorted array				
unsorted array				
hashtable				
sorted linked list				
unsorted linked list				

Questions on data structures

Q1. (a) In the class below, insert a static procedure that appends to linked list L1 the elements of L2 –the last node of L1 is changed to contain the name of the first cell of L2. The method does not return anything. The lists are built using class ListNode, given below.

(b) What is the asymptotic complexity of your algorithm, expressed as a function of n1 and n2, where n1 is the number of elements in L1 and n2 is the number of elements in L2? Justify your answer (briefly).

```
class ListNode {
  protected Object element;
  protected ListtNode next;

  // Constructor: instance with elem. x , next field n
  public ListNode (Object x, ListNode n)
    { element = x;  next = n; }

   public Object getElement ()
     { return element;  }

  public ListNode getNext ()
    { return next;  }

  // Set this node's element to x
  public void setElement (Object x)
    { element = x;  }

  // Change this next field to n
  public void setNext (ListNode n)
    { next = n; }

  // = elements of this list, separated by " "
  public String toString () {
    if (next == null)
        return element.toString();
    return element.toString() + " " + next.toString();
  }
}
```

Q2. You have a hash table of size m=11 and a (not very good) hash function h:

h(x) = (sum of the values of the first and last letters of x) mod m

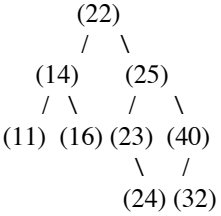
where the value of a letter is its position in the alphabet (e.g., value(a)=1, value(b)=2, etc.). Here are some precomputed hash values:

ape, 6	bat, 0	bird, 6
carp, 7	dog, 0	hare, 2
ibex, 0	mud, 6	koala, 1
stork, 8		

Draw a picture of the resulting hashtable (using linear probing) after inserting, in order, the following words: *ibex, hare, ape, bat, koala, mud, dog*. Which cells are looked at when trying to find *bird*? What is the load factor for the hash table?

Q3. (a) What is quadratic probing, and why is it preferred over linear probing?
(b) What is a probe? What is the expected number of probes, using linear probing, when the load factor is 1/2?

Q4. Where is the largest element in a binary search tree (BST)? The following picture represents a BST:



(a) In what order are the nodes visited during a preorder traversal? Inorder traversal? Postorder traversal?
(b) Show the tree that results when 15 is inserted into the tree.

Q5. Write a method that takes as input a binary search tree T and two keys k1 and k2, which are ordered so that k1 <= k2, and prints all elements X in the tree such that k1 <= key(X) <= k2. You may assume that the keys are of type Comparable. Your program should run in time O(K + log N), where K is the number of keys printed.

Q6. Write recursive methods for printing the inorder, preorder, and postorder list of the nodes of a binary tree.

Q7. For each of the following problems, choose the best of the listed data structures and explain why your choice is best. Assume that the data set is going to be large unless otherwise indicated. Where several operations are listed you should assume, unless stated otherwise, that the operations occur with about equal frequency.

(a) The operations are Insert, DeleteMax, and DeleteMin: sorted array or *sorted doubly-linked list*

(b) You have a set containing Java's keywords: *ordered array* or *balanced tree*

(c)You have a set that can contain anywhere from 100 to 10,000 words. Operations are insert, find: *unordered linked-list* or *hash table*

(d) You have a large set of integers with operations insert, findMax, and deleteMax: *unordered array* or *Hashtable*

Questions on older material (classes, subclasses, interfaces, exceptions)

Q1. Explain in a sentence or two the difference be-

tween overloading and overriding.

Q2. Consider a database for keeping track of student grades in CS211. A student's scores in the assignments are kept in an object of type StudentGrades; array element Scores[i] is the score in assignment i for that student. An object of type BigBook contains an object of type StudentGrades for every student who has submitted an assignment. The total number of students in the course is known to be less than 100. An outline of the class definitions are shown below.

```
class BigBook {
  private int MaxClassSize = 100;
  private int numAssignments; // No. of assignments
  private studentGrades[] Folio;  // An entry is either
       // null or the name of a StudentGrades folder

  // Constructor: an empty folio of students and
 // NA assignments
  public BigBook(int NA) {
       numAssignments= NA;
       Folio = new StudentGrades[MaxClassSize];
  }
}

class StudentGrades {
  private String name;  // Scores contains the grades
  private int[] Scores;  // for students name

  // Constructor: a student named grunt with an
  //   array of NA assignment scores
  public StudentGrades(String grunt, int NA)
    { name = grunt;  Scores = new int[NA]; }
}
```

(a) Write methods in one or both classes that will permit my secretary to enter a grade for a student for a particular assignment into her database. The method invocation must look like the following:

```
CS211database.Enter(``Albert Gore", 4, 10);
```

In this invocation, CS211database is an object of type BigBook, and Enter is an instance method. This invocation should check the database for a student named Albert Gore. If this student is not in the database, first insert them into the database. Then, change the grade for assignment 4 with a grade of 10, provided the assignment number is in the range 0..NumAssignments-1. Do not add or remove any instance variables in either class.

(b) If a method in class StudentGrades is passed a reference to an object of type BigBook, can that method access the field NumAssignments in this object? Explain two ways in which this method can determine the number of assignments.

Q3. (a) Write exactly one method that takes in an integer and may throw one of two new exceptions (of type Exception) with messages "Too high!" and "Too low!". If the value is out of the range 0..100 (inclusive), throw the appropriate exception.

(b) Suppose method A calls method B, method B calls method C, and C is defined to potentially throw an exception. Can this exception be handled in method A? If not, why? If so, how?

Q4. Java does not allow multiple inheritance because two classes can define methods with the same signature that do completely different things. Therefore, if a child class extends two such classes, if it refers to one of these pre-existing methods in a super class, JAVA won't know which method to execute. However, Java does allow a class to implement more than one interface. Why don't interfaces have the same problem with method ambiguity that classes do?

Questions on recursion

Q1. Give a recursive algorithm for making change with the smallest number of coins. The input to your method is an integer array coins[] containing the possible values for the coins and an int that is the change needed. The output should be an array indicating the number of each coin needed. So, if the input is coins[] = {1, 5, 10, 25} and amount = 55, then the output should be an array change[] = {0, 1, 0, 2}. Hint: For the recursive step, subtract the amount of one coin from the amount of change needed. Be careful to use the smallest number of coins in your solution.

When you are finished, try your program with coins[] = {1, 5, 21, 25} and amount 63. This should show you that the obvious way of computing the change, the greedy algorithm that uses as many high coins as possible, does not always deliver the optimium. In this case, three 21-cent coins is best.

Q2. Specify and a recursive method that determines whether a String s is a palindrome. A palindrome is a String that reads the same backward and forward.