UNIVERSITY OF WATERLOO

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 250 ALGORITHMS AND DATA STRUCTURES

Final Examination (8 pages) Instructor: R.E.Seviora 9-12 AM, Apr 16, 2003

Name (last, first)			Student ID			Total
1.	2.	3.	4.	5.	6.	7.

Do all problems. The number in brackets denotes the problem weight (out of 100). If information appears to be missing from a problem, make a reasonable assumption, state it and proceed. If the space to answer a question is not sufficient, use the last page. Closed book. Only numerical display calculators allowed.

PROBLEM 1 [16]

A. Recurrences

After analyzing the running time of an algorithm, you obtained the following recurrence

$$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1)+1 & n > 0 \end{cases}$$

Solve this recurrence.

B. Asymptotic Bounds

1. Company X which makes supercomputers claims that their latest computer will run 100 times faster than the fastest computer of competitor Y.

   If Y's computer can execute a program of size N in one hour, what size input can X's computer execute in one hour for program with the growth rates given in the table below. (For example, if the program running on X could execute in one hour a problem which is twice as large as that which would execute on Y in one hour, the answer in the rightmost column would be 2N.)

Program	*$\Theta()$*	*Size of Problem Solved in 1 hour on X's computer*
A	n	
B	n^2	
C	n^3	
D	2^n	

2. Assume that the array `a1` contains a random permutation of the values of 0 to (n-1). (For example, if n=4, `a1` could contain {0, 2, 3, 1}, or {2, 0, 3, 1}, etc.). For the following program fragment, determine the asymptotic theta bound for the running time for the average case:

```
int sum = 0;
for (i=0, i<n, i++) {
  for (j=0; a1[j]!=i; j++)
    sum++;
}
```

PROBLEM 2 [16]

A. Queues

Consider the class `QueueAsArray` discussed in the course. You were given the following implementation of the method `enqueue` and asked to confirm that it does properly implement the enqueue operation. State whether this is the case or not. If your answer is no, show the modifications required.

```
public class QueueAsArray extends AbstractContainer implements Queue {
  //fields
  protected Object[] array;
  protected int head;
  protected int tail;
  //methods
  public void enqueue (Object o) {
    if (count == array.length)
      throw new ContainerFullException();
    array[++tail] = o;
    count++
  }
  ..
}
```

B. Ordered/Sorted Lists

The class `OrderedListAsArray` discussed in the course implemented the interface `SearchableContainer`, which included methods `boolean isMember(Comparable obj)` and `Comparable find(Comparable obj)`. Its class definition had just one field, `Comparable[] array`.

Consider the following program fragment, where `Int` is the course-defined wrapper class for `int`'s:

```
Comparable a = new Int(3);
Comparable b = new Int(6);
Comparable c = new Int(6);
Comparable d = new Int(6);
OrderedList l = new OrderedListAsArray(4);
l.insert(a);
l.insert(b);
l.insert(c);
```

(i) Show the contents of the ordered list `l` (i.e. the contents of the object the variable `l` refers to etc.) after the execution of the above fragment. Make sure you show all the objects.

`l` 

(ii) State what the variables `b` and `p` will contain after the execution of the following statements:

```
boolean b = l.isMember(d);

Comparable p = l.find(d);
```

2. Consider the two implementations of sorted lists presented in the course. For each implementation, show the tight big-Oh bounds on the running times of the following methods.

Method	*SortedListAsArray*	*SortedListAsLinkedList*
`insert`		
`isMember`		
`find`		
`withdraw`		
`get`		
`findPosition`		
`Cursor.getDatum`		
`Cursor.withdraw`		

PROBLEM 3 [14]

A. Hash/Scatter Tables

1.Consider an open addressing scatter table of length 11 and h(x) = x mod 11. You are to insert keys {17, 38, 35, 80, 46, 03, 72, 79} into the tables in the order given, for the two collision resolution strategies listed below. In each case, show the contents of the table after all items have been inserted. If a strategy does not permit such insertion, explain why not.

(i) Open addressing with *linear probing.*

(ii) Open addressing with *quadratic probing.*

x	*h(x)*
17	6
38	5
35	2
80	3
46	2
03	3
72	6
79	2

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

(i)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

(ii)

2. A partly filled chained scatter table is shown below (label 'before'). Here, h(x) = x mod 11 and the entries in the table hash to the values shown at the left. In the table labeled 'after', show the contents of the chained scatter table after the withdrawal of the key 59 .

x	h(x)
05	5
15	4
28	6
39	6
45	1
59	4
67	1
70	4
96	8

(ordered by x)

0	null	nil
1	67	02
2	45	nil
3	null	nil
4	59	05
5	15	06
6	28	07
7	39	08
8	05	09
9	96	10
10	70	nil

(before)

0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

(after)

B. Hash/Scatter Tables: Running Time and Memory

1. In the course, we considered open addressing scatter tables with lazy withdrawal. The (worst case) running time for `find` was expressed as `T<hashCode> + nT<isEq> + O(M)`. Here, `M` denotes the size of the table and `n` the number of items stored in the table.

Isn't this wrong? The running time of `find` should depend on `n`, but not on `M`. Shouldn't `O(M)` be replaced by `O(n)`? Justify your answer.

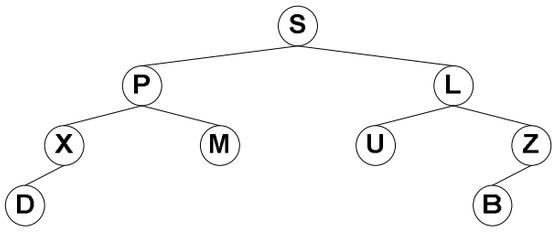
2. Consider the class `ChainedHashTable` discussed in the course. This class implemented the hash table with separate chaining. Its class definition contained just one field `LinkedList[] array`. The class `LinkedList was` introduced earlier in the course.

   Let `M` denote the length of the array in the table and `n` the number of items in the table. Give an expression for the memory requirements of such table as a function of `M` and `n`. Use `Sref` and `Sint` to denote the size of object reference and `int`, respectively. Draw a figure to illustrate.

PROBLEM 4 [14]

A. Tree Traversals

1. Consider the binary tree shown below. For each of the traversals listed, give the order in which the nodes are visited.



preorder									
inorder									
postorder									
breadthfirst									

2. Construct the expression tree whose postorder traversal results in `ayz*+p/kbc-*+`.

B. Space Requirements of Trees

1. The course discussed several implementations of trees, including
   (a) the class `NaryTree` which extended `AbstractTree` (which in turn extended `AbstractContainer`). Its class definition had fields `Object key; int degree; NaryTree[] subtree;`
   (b) the class `BinaryTree`, which also extended `AbstractTree`. Its class definition contained three fields - `Object key; BinaryTree left; BinaryTree right`.

   State the space occupied by *one* (non-empty) instance of
   (i) `NaryTree` (use the letter `N` to designate the degree of the tree)

   (ii) `BinaryTree`

PROBLEM 5 [16]

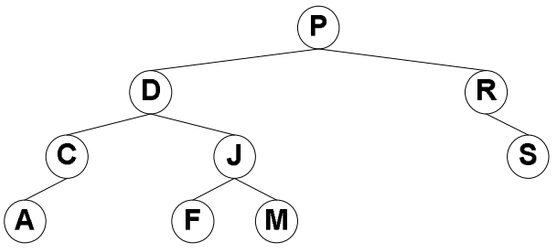
A. AVL Trees

1. What is the minimum and the maximum number of internal nodes in an AVL tree of height three? Draw a concrete example of an AVL tree which illustrates the *minimum* case.

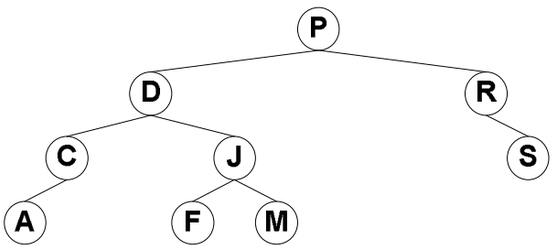
2. Consider the AVL tree below. A new key (H) is to be inserted.

   (a) In the tree below, show where the node H would be inserted *before* balancing.

   (b) Identify the rotations required, if any. In the empty space on the right, show the tree *after* balancing.



3.Consider the AVL tree below. The key S is to be withdrawn. Identify the rotations required, if any. In the empty space on the right, show the resultant tree *after* balancing.

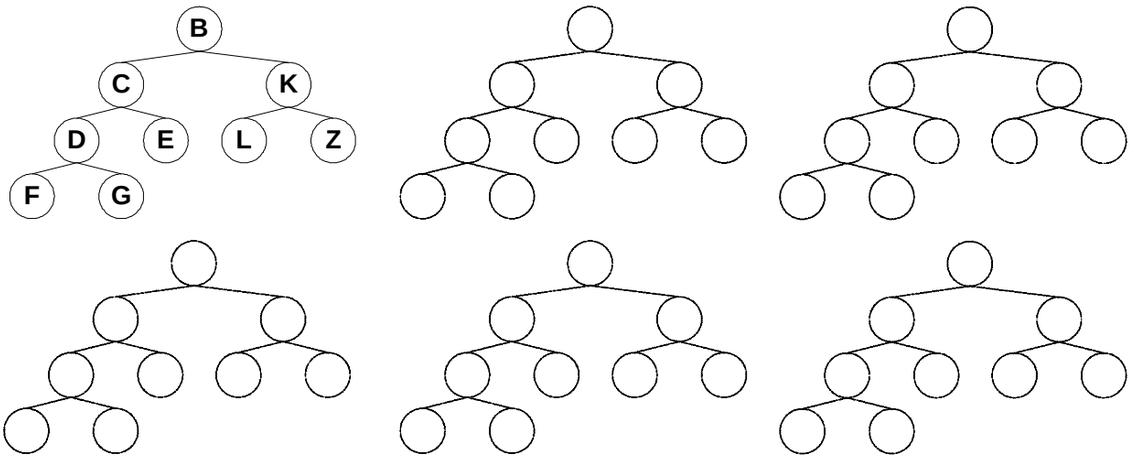


B. Binary Heaps

1.Consider an array implementation of a binary min heap in which all of the keys are single letters and lexicographic order is used (i.e. A<B<C...). The heap is initially empty. The items {G, E, C, B, F, D, A} are to be inserted into the heap, in the order given. Complete the table below by showing the array contents after each insertion.

item	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
G							
E							
C							
B							
F							
D							
A							

2.Consider the binary min heap given below. Using the tree outlines given below, show the steps in the execution of `dequeueMin`. (If appropriate, you may add additional trees or not use all of the outlines shown.)



PROBLEM 6 [8]

A. Project 4

In project 4, you were to design and code the class `UnboundedLife`. The class had to implement the interface `GameOfLife` which had had three main methods, `void setOccupied (int x, int y, boolean occupied)`, `boolean isOccupied(int x, int y)`, and `void nextGeneration()`. During the design, you were to consider several candidate data structures for storing the state of the game (i.e. the occupied cells), analyze their suitability and select the most appropriate one.

The table below shows several data structures discussed in the course. For each structure, state the tight big-Oh bound for the average and the worst case running time for the methods given in the header, as a function of the number of live cells, `n`, in the game. If the data structure uses keys, consider keys that are objects wrapping `long`, whose upper 32-bits is the `x` coordinate and lower 32 bits is the `y` coordinate.

Data Structure	*O( ) for isOccupied(x, y)*		*O( ) for setOccupied (x, y, true)*	
	average case	*worst case*	*average case*	*worst case*
LinkedList				
SortedListAsLinkedList				
ChainedHashTable				
OpenScatterTable				
BinarySearchTree				
AVLTree				

If appropriate for an answer above, state the assumptions under which it applies.

PROBLEM 7 [16]

A. Greedy Algorithms

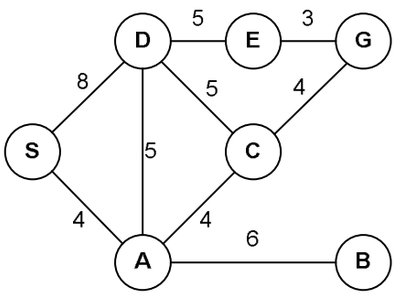
Consider the scale balancing problem discussed in the course. In this problem, a set of weights is to be placed on the pans of the scale so that the scales are balanced or the imbalance is minimized. Devise a greedy algorithm to solve it. Describe its steps. State clearly what the algorithm is 'greedy by'.

2. Consider a concrete example of this problem with four weights {7, 6, 5, 4}. For the algorithm you devised in step 1, show the sequence in which the weights would be placed and the pan they would be placed on, for example { …., 6 to R, 7 to L,…} Here L denotes the left and R denotes the right pan .

B. Backtracking Algorithms

The map below shows seven cities and connecting roads. The road distances are also given. The goal is to find the shortest route from city S to city G.

Devise a backtracking algorithm for solving this problem. Show the solution space. To facilitate the marking of your answer indicate clearly the partial route and the distance accumulated in each node of the solution space (e.g. [SDE, 13]).



OVERFLOW SHEET [Please identify the question(s) being answered.]