UNIVERSITY OF WATERLOO

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 250 - ALGORITHMS AND DATA STRUCTURES

Final Examination Instructor: R.E.Seviora December 9, 1999, 2-5 PM

Name:		**Student id:**	
1	2	3	4
5	6	7	Total:

Do **all** problems.The number in brackets[..] denotes the relative weight of the problem (out of 100). If information appears to be missing from a problem, make a reasonable assumption, state it and proceed. If the space for an answer is not sufficient, use the last (overflow) page. Closed book; no calculators.

PROBLEM 1 [10]

A. Asymptotic bounds

i. Give a definition for the *tight* big Oh bound (for a non-negative function f(n), $n\geq 0$).

ii. For two functions, f(n) and g(n), define the relation '$\leftarrow$' by $f(n) \leftarrow g(n)$ if $f(n) = O(g(n))$. For example, $2n \leftarrow (n^2+1)$. Arrange the following functions in the '$\leftarrow$' order:

n^2, 2^n, $\log n$, $n\log n$, 3^n, $(\log n)^2$, n, $n/\log n$, $n(\log n)^2$.

B. Solving recurrences

In a number of cases the running time of a recursive algorithm can be expressed by the following recurrence:

$$T(1) = 1$$
$$T(n) = aT(n/2) + n^k, \quad n>1.$$

where a and k are integers, $a>0$, $k\geq 0$. This is the case, e.g., with certain divide-and-conquer algorithms. Solve this recurrence. You may assume $n=2^m$.

PROBLEM 2 [12]

A. Ordered list, append

i. Devise an algorithm to append the contents of one ordered list to the end of another. Both lists are represented using linked lists.

ii. What is the running time of your algorithm?

```
public class OrderedListAsLinkedList
  extends AbstractSearchableContainer implements OrderedList  {
  protected LinkedList linkedList;
  public void insert (Comparable object) {
    linkedList.append (object); ++count; }
  //...
  public void append (OrderedListAsLinkedList list) {
```

B. Ordered list, running time

Consider an implementation of the OrderedList which uses a doubly linked list (i.e. each list element contains a reference to the immediately preceding and the immediately following element on the list). Compare the running times of the OrderedList operations for this implementation with the one which uses singly linked list given in the table below. Briefly explain the differences, if there are any.

Method	singly-linked	doubly-linked
`isMember`	O(n)	
`insert`	O(1)	
`find`	O(n)	
`withdraw`	O(n)	
`findPosition`	O(n)	
`Cursor.getDatum`	O(1)	
`Cursor.insertAfter`	O(1)	
`Cursor.insertBefore`	O(n)	
`Cursor.withdraw`	O(n)	

PROBLEM 3 [18]

A. Hash function characteristics

List three key characteristics of a good hash function.

B. Hash/scatter tables

The following set of keys is to be inserted in an initially empty table. The table size is 13_{10}. Show the contents of the table after the keys have been inserted, in the order douze, onze, ..., deux, un, for the following cases:

i. chained hash table

ii. open scatter table using linear probing.

x	$h(x)_{10}$
"douze"	37
"onze"	37
"dix"	56
"neuf"	38
"huit"	52
"sept"	52

x	$h(x)_{10}$
"six"	56
"cinq"	49
"quatre"	37
"trois"	51
"deux"	56
"un"	46

chained hash table

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

scatter table (linear prob)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

C. Hash/scatter tables, time-space tradeoffs

Running time analyses suggest that chained hash tables have superior running times when compared to scatter tables. They, however, need more memory. Derive an expression for the total memory space needed for a table of size M with n elements. Consider the two cases given below. You may assume that all integers and object references require 4 bytes.

i. chained hash table

ii. scatter table (open addressing)

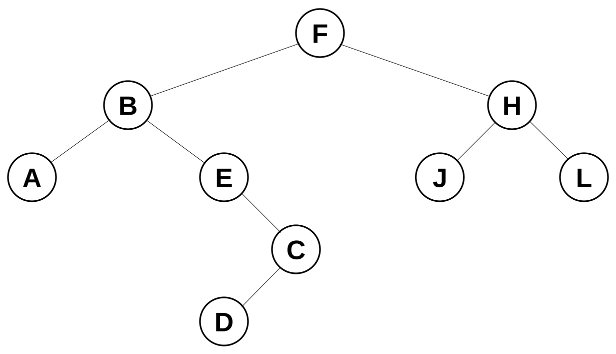
PROBLEM 4 [16]

A. Trees

Give a mathematical definition of a tree. (You can give either the 'set' definition presented in the text/lectures or, alternatively, a 'relational' definition that is also used.)

B. Tree traversal

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



preorder									
inorder									
postorder									
breadth-first									

ii. Construct the expression tree whose PostOrder traversal is: `22^a*b*b2^+ab-/`. Here, `^` denotes the binary operation of exponentiation and `+`, `-`, `*`, `/` have their usual meanings.

C. Binary expression trees, postfix printout

The program shown below defines a visitor that prints the expression represented by an expression tree in ***infix*** notation. Modify it so that it prints the expression in ***postfix*** notation.

```
public class ExpressionTree
  extends BinaryTree
{ public String toString () {
  final StringBuffer buffer
    = new StringBuffer();
  PrePostVisitor visitor
    = new AbstractPrePostVisitor () {
    public void preVisit (Object obj)
      { buffer.append ("("); }
    public void inVisit (Object obj)
      { buffer.append (obj); }
    public void postVisit (Object obj)
      { buffer.append (")"); }
  }
  depthFirstTraversal (visitor);
  return buffer.toString ();
  }//...
}
```

PROBLEM 5 [16]

A. Greedy algorithms

Devise a greedy algorithm to solve the scales balancing problem discussed in the course. State clearly what the algorithm is 'greedy by'.

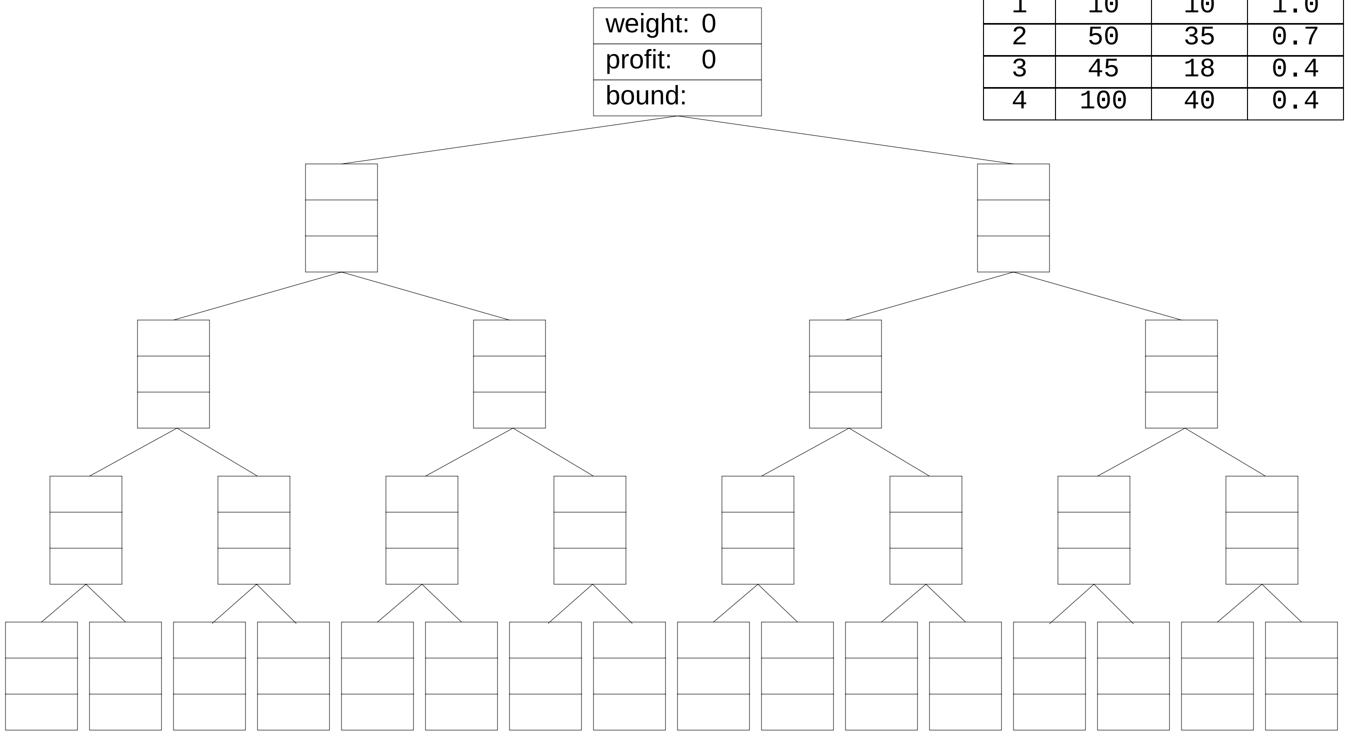
B. Branch-and-bound algorithms

The 0/1 knapsack problem: In the lecture, we considered two solutions to this problem - greedy and bottom up. A backtracking algorithm can also be devised for the solution of the knapsack problem. The performance of this algorithm can be further improved by defining a suitable bounding function and using it to reduce the number of nodes explored.

i. Define a suitable bounding function

ii. Consider the case when four items with the weights and profits shown below are available. The total carrying capacity of the knapsack is 100. Show the solution space. If needed, add other nodes to the tree. Mark each node with the total weight and profit accumulated to that point and the bound value. Cross out the nodes which need not be explored. (Observation: establish first the order in which items are considered for inclusion in the backpack.)

item	weight	profit	prof.den
1	10	10	1.0
2	50	35	0.7
3	45	18	0.4
4	100	40	0.4

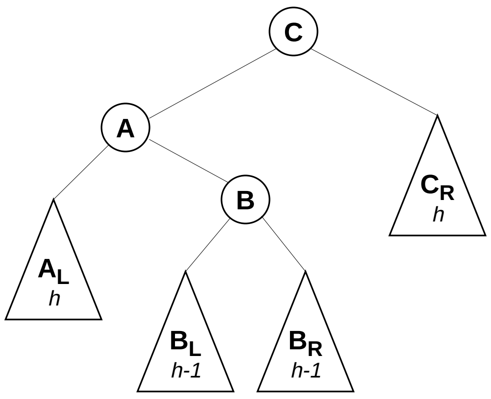


PROBLEM 6 [12]

A. AVL trees, rotation

Under some circumstances, the `insert` method for AVL trees must do a double rotation to restore the AVL property. One such circumstance applies when the item inserted would increase the height of the B_R subtree by 1 (see the figure below; subtree height shown in italics).

Show the rotation required and state why the resulting tree satisfies the AVL property.



B. AVL trees, construction

Draw the sequence of AVL trees obtained when the following keys are inserted, one-by-one, in the order given into an initially empty AVL tree: `{F,E,B,A,C,D,G}.` If the rotation considered above needs to be carried out during construction, identify it.

PROBLEM 7 [16]

A. Sorting

Draw a sequence of diagrams that trace the execution of the following sorting algorithms.

i. Straight insertion sort.
ii. Heapsort.

i. Insertion sort

4	**2**	**5**	**2**	**6**	**8**	**3**	**7**	**6**	**5**

ii. Heapsort

1	2	3	4	5	6
F	**C**	**D**	**A**	**B**	**E**

B. Running times for sorting

Determine big-Oh running times for the following two cases. In both cases, state what the (best/worst) case considered is. The big-Oh bound should be reasonably tight.

i. insertion sort: the best case
ii. quicksort: the worst case.

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