Carleton UNIVERSITY

FINAL
EXAMINATION
FALL 2006

DURATION: 3 HOURS **No. Of Students: 20**

Department Name & Course Number: **Systems and Computer Engineering**
SYSC 2100

Course Instructor (s): Thomas Kunz

AUTHORIZED MEMORANDA
NONE

Students MUST count the number of pages in this examination question paper before beginning to write, and report any discrepancy to a proctor. This question paper has ____7____pages + cover page = _8__ pages in all.

This examination question paper may not be taken from the examination room.

In addition to this question paper, students require: an examination booklet no
Scantron Sheet no

Name: ____

Student Number: ____

Question 1: Recursion (10 marks)

Recursively define the following types of binary trees, based on the non-recursive definitions used in class:

1) Full binary tree
2) Complete binary tree
3) Balanced binary tree

Question 2: Linked List (10 marks)

1) The last node of a linear linked list ______.

    a) has the value `null`

    b) has a `next` reference whose value is `null`

    c) has a `next` reference which references the first node of the list

    d) cannot store any data

    Answer:

2) Which of the following will be true when the reference variable `curr` references the last node in a linear linked list?

    ```
    a) curr == null
    b) head == null
    c) curr.getNext() == null
    d) head.getNext() == null
    ```

    Answer:

3) If a linked list is empty, the statement `head.getNext()` will throw a(n) ______.

    ```
    a) IllegalAccessException
    b) ArithmeticException
    c) IndexOutOfBoundsException
    d) NullPointerException
    ```

    Answer:

4) What are two advantages of using a reference-based implementation of the ADT list instead of an array-based implementation?

5) Write the code fragment to delete the node that the reference variable `curr` references in a circular doubly linked list? Each node supports the methods `setNext()`,`getNext()`,`getPrecede()` and `setPrecede()`.

6) Write the code fragment to insert a new node that the reference variable `newNode` references before the node referenced by the reference variable `curr` in a doubly linked list, using the methods listed in part 5).

Question 3: Algorithm Complexity (5 marks)

1) Assuming a linked list of n nodes, the code fragment:

```
Node curr = head;
while (curr != null) {
   System.out.println(curr.getItem());
   curr.setNext(curr.getNext());
}  // end while
```

requires ______ assignments.

   a) n
   b) n – 1
   c) n + 1
   d) 1

   Answer:

2) Assuming a linked list of n nodes, the above code fragment (in part 1)) requires ______ comparisons.

   a) n
   b) n – 1
   c) n + 1
   d) 1

   Answer:

3) Assuming a linked list of n nodes, the above code fragment (in part 1)) requires ______ write operations.

   a) n
   b) n – 1
   c) n + 1
   d) 1

   Answer:

4) Consider an algorithm that contains loops of the form:

```
for (x = 1 through n ) {
  for (y = 1 through x) {
    for (z = 1 through 10) {
      Task T
    }  // end for
  }  // end for
}  // end for
```

If task T requires t time units, the innermost loop on z requires ______ time units.

   a) y
   b) 10
   c) z * t
   d) 10 * t

   Answer:

5) Consider the above algorithm (part 4) again.
If task T requires t time units, the loop on y requires ______ time units.

   a) 10 * t
   b) (10 * t) + x
   c) 10 * t * x
   d) t * x

   Answer:

Question 4: Binary Trees (10 marks)

1) Beginning with an empty binary search tree, what binary search tree is formed when you insert the following values in the order given?
    a. W, T, N, J, E, B, A
    b. W, T, N, A B, E, J
    c. A, B, W, J, N, T, E

2) Write pseudocode for a method that performs a range query for a binary search tree. That is, `RangeQuery(tree, low, high)` should visit all items/nodes in `tree` that have a search key `k` with `low <= k <= high`. To indicate the a specific node `x` is being visited (printed, modified, …), simply use the pseudocode instruction `visit(x)`.

3) Proof by induction that a binary tree with *n* nodes has exactly *n+1* empty subtrees (or, in Java terms, *n+1 null* references in a reference-based implementation)

Question 5: Heaps/Priority Queues (10 marks)

1) Given the following `maxheap h,` show what the heap would look like after each of the following pseudocode operations:

```
        13
       /  \
     10    9
    / \   /
   3   7 8
```

   a) `h.heapInsert(16)`
   b) `h.heapInsert(14)`
   c) `h.heapDelete()`

2) Does the order in which you insert items into a heap affect the heap that results? Explain.

3) Suppose that after you have placed several items into a priority queue, you need to adjust one of the priority values. For example, a particular task in a priority queue of tasks could become more or less urgent. How can you adjust the heap if a single priority value changes? Note: your solution should be better than "remove the item and re-insert it" as we have no operation to delete arbitrary items from a heap.

Question 6: Balanced Trees/Tables (10 marks)

1) What are the advantages of implementing the ADT table with a 2-3 tree instead of a binary search tree? Why do you not, in general, maintain a completely balanced binary search tree?

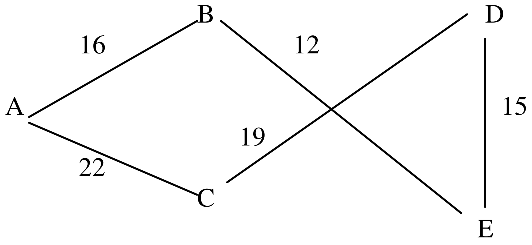
2) Given the 2-3 tree below, draw the tree that results after inserting k, b, c, y, and w into the tree.

```
                 [r]
               /      \
       [e h]            [u]
       /  |  \         /    \
    [d] [f] [o p]   [s t]    [v]
```

3) Write pseudocode for the *tableDelete* operation when the implementation uses hashing and linear probing to resolve collisions.

Question 7: Graphs (10 marks)

1) Use both the depth-first and the breadth-first strategy to traverse the graph below, beginning with vertex A. List the vertices in the order in which each traversal visits them. If a node has multiple neighbors, assume that they are visited in alphabetical order



2) By modifying the DFS traversal algorithm, write pseudocode for an algorithm that determines whether a graph contains a cycle.

3) For the graph above, draw all possible spanning trees. Which one is the minimum spanning tree?