Information on Midterm #2

Date: Friday, May 21, 2009
Type: closed book exam.

Below are the topics covered in this exam. For each topic I mentioned the chapter and the section in the textbook where you can find the related material. I recommend you take as a guide the lecture handouts I provided for you since in the book there are certain things that we didn’t cover, or covered a little differently, and there are certain details/topics that are NOT covered in the book (e.g. implementing an iterator for different tree traversals, selection and bubble sorts).

You are expected to know all the material covered in class exactly the was it was covered). Any material discussed/covered in lectures, labs, and projects can be included in the test; there won’t be anything in the test that was not covered in a lecture, lab, or project.

Topics:

1. Binary tree traversals (Chapter 4, sections 4.1.2 and 4.6):
   - Know all 4 types of binary tree traversal strategies and be able to list elements stored in the given binary tree using each of the traversal strategies.
   - Know how to implement an iterator to give the client ability to traverse the binary tree, e.g. a BST (see Project 3); be able to write all relevant pieces of Java code for the implementation of an iterator (for pre-order, in-order, and level-order traversals), be able to write client code that uses an iterator to traverse a collection stored in a binary tree (for example in a BST).
   - For a given binary tree be able to illustrate the work of consecutive next operations of the iterator class (as done during the lecture): show the initial content of the stack/queue, and its content after each next operation until the whole tree is traversed (pre-order, in-order, and level-order traversals only)

2. Hash Tables (Chapter 5, sections 5.1-5.5):
   Know the main concepts of a hash table arrangement. Know what a hash function is and how to compute it for different types of keys: (i) for integer keys, (ii) for string keys. Be able to implement (code) the private hash method inside a class implementing a Hash Table ADT.
   a) Hash Tables with separate chaining.
      - Be able to give the implementation details of a class defined for such tables – know what the instance variable should be, what the constructor should do.
      - Be able to implement (write Java code) basic operations: insert, delete, find (private hash method included), as well as other operations on a collection (see Labs 12 and 13).
      - Be able to illustrate the work of consecutive insert operations on a given example.
      - Be able to implement an iterator class for hash table with separate chaining (see Lab13).
      - Be able to give the running time estimates for basic operations.
   b) Hash Tables with open addressing
      - Know the problems of this type arrangement: agreements/rules for table size, rehashing.
      - Be able to give the implementation details of a class defined for such tables – know what inner classes you need, what instance variables should be, what the constructor should do.
      - Be able to implement basic operations: insert, delete, find (private methods findposition, hash, and rehash included), as well as other operations on a collection (see Project 4).
- Be able to illustrate the work of consecutive insert operations on a given example using (i) linear probing, and (ii) quadratic probing.
- Be able to implement an iterator class for hash table with open addressing (see Project 4).
- Be able to give the running time estimates for basic operations.

3. Exercise on Recursion:
Be able to implement a recursive routine (similar to the recursive methods you have implemented in Labs 8, 9, 11, 15, and Project 3).

4. Simple sorting algorithms (Chapter 7, sections 7.1 - 7.3)
For **bubble sort** and **insertion sort** algorithms:
- Be able to illustrate the work of each algorithm on a given list (show the content of the list at any point during algorithm’s work).
- Be able to write Java code to implement each algorithm.

For each of 3 simple sorting algorithms: **selection sort**, **bubble sort**, and **insertion sort**:
- Be able to give the order of the algorithm.
- Be able to identify the best, worst and average cases, give and explain the running time estimates for each of these cases.

**In regards to simple sorting techniques:**
- Know what an inversion is.
- Be able to recognize/point out all inversions in the given list.
- Know the average number of inversions in a list.
- Know the **lower bound** for Simple Sorting Algorithms

**Lecture handouts related to the material covered in Midterm 1:**
- Traversing a Binary Tree
- Iterators
- Traversing a Binary Tree with an Iterator
- Hashing
- Hash Tables with Separate Chaining
- Hash Tables with Open Addressing
- Sorting
- Insertion Sort (examples)

**Supplementary handouts containing sample codes** for the parts of following assignments:
Lab8 (recursive routines of all 3 exercises)
Lab9 (MyList class)
Lab11 (BST operations) and Projec3 (print)