CSE250, Spring 2013 Prelim II Apr. 15, 2013

Closed book, closed-notes-except-for-1-sheet, closed neighbors, 48 minutes. Do ALL THREE questions on the exam pages themselves. Please *show all your work*; this may help for partial credit. The exam totals 67 pts., subdivided as shown.

(1) (6 + 4 + 4 + 4 + 6 + 4 = 28 pts. total)

First, let us employ the simple hash function h that adds up the index-numbers of all letters in alpha-order a = 1, b = 2, c = 3, etc. For instance, h("bud") = 2+21+4 = 27, which gives 3 mod 8.

(a) For a hash table with chaining of size 8 that is behaving like a set, insert the four words `bad bed bid dad` in that order.

(b) Now iterate thru the hash table from bucket 0 down to 7, and reinsert the items in *that* order into a hash table of size 4.

(c) Take the same iteration in (b) and insert the words into a binary search tree, using alphabetical order as the comparison function.

(d) This time show the BST that results from the insertion sequence `bad bed bid dad` as in (a).

(e) Now show the steps and result of forming a heap out of the length-4 vector `bad bed bid dad`, keeping the maximum word in alphabetical order on top. You may use either the technique of inserting the words in that order into an initially empty heap, or the generally-faster results of calling `make_heap` on the vector as given.

(f) Comparing your answers to (c) and (d), and thinking of all the data structures in this problem, suppose you had a longer vector in sorted order like `bad bed bid dad did eat fat goose...` How would you try to make a well-balanced binary search tree out of that vector?

(2) ($9 \times 3 = 27$ pts. total)
For each task below labeled 1.–10., say which of the following best describes its running time:

(a) Guaranteed $O(1)$ time.

(b) Amortized $O(1)$ time.

(c) Usually $O(1)$ time.

(d) Guaranteed $O(\log n)$ time.

(e) Usually $O(\log n)$ time.

(f) Guaranteed $O(n)$ time.

In all cases n denotes the number of items currently in the underlying data structure, and any other parameters are stated. The variable `vec` stands for a vector, `dlist` for a doubly-linked list (not necessarily sorted), `valli` for a "`Valli`" data structure, `bst` for a BST—i.e. a general binary search tree, `itr` for an iterator of the appropriate kind, and `item` for a typical item in the data structure. *Justifications* are not required, but might help for partial credit. The tasks are:

1. For a BST iterator `itr`, the call `itr++;` _____

2. `dlist.push_back(item);` (equivalently, `dlist.insert(dlist.end(),item);`) _____

3. `vec.push_back(item);` _____

4. `dlist.erase(itr);` _____

5. `vec.erase(itr);` where `itr` points in the middle of `vec`. _____

6. For a Valli iterator `itr`, the call `itr++;` _____

7. `valli.find(item);`, assuming `valli` just refreshed with `ratio` $r = \log_2(n)$. _____

8. `bst.find(item);` _____

9. Preorder traversal of a BST. _____

(3) (12 pts.)

Suppose Project 1 had been changed to make you create a set of words that do **not** remain words when "turned inside out." That is, suppose you have the following containers:

```
vector<string> words;
Valli<string> turnedWords;
...
//assume you've already run code that reads the dictionary file into words
//and for every read word, inserts wordTurn(word) into the Valli container.
...
set<string> unturnables;   //initially the empty set
...
//your code on this problem goes here.
```

Write code with iterators that inserts into `unturnables` every word that is *not* in `turnedWords`—that is, not equal to the turning of itself or some other word. End of Exam