CS 2150 final exam

Name _____

You MUST write your e-mail ID on **EACH** page and bubble in your userid at the bottom of this first page. And put your name on the top of this page, too.

If you are still writing when "pens down" is called, your exam will be ripped up and not graded – even if you are still writing to fill in the bubble form. So please do that first. Sorry to have to be strict on this!

Other than bubbling in your userid at the bottom of this page, please do not write in the footer section of this page.

There are 10 pages to this exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts of points.

If you do not bubble in this first page properly, you will not receive credit for the exam!

This exam is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

_____

_____

_____

_____

_____

You step in the stream,
But the water has moved on.
This page is not here.

(the bubble footer is automatically inserted into this space)

Page 2: C++

1. [3 points] What is the difference between shared multiple inheritance and replicated multiple inheritance? When would you want to use each?

2. [3 points] When would you want to use private inheritance in C++?

3. [3 points] List two new features in C++11. If it's not obvious from their name, briefly describe said features.

4. [3 points] For each of the two locality properties of caches, *why* do they make caches work well?

Page 3: Trees and Hashes

5. [6 points] Give one advantage and one disadvantage of each of the three types of balanced binary trees: AVL, red-black, and splay.

6. [6 points] Give one advantage and one disadvantage of each of the four types of collision resolution that we studied in class: separate chaining, linear probing, quadratic probing, and double hashing.

Page 4: Multimaps

[8 points] A *map* is an abstract data type that allows for key-value pairs. We've seen maps with both trees (the sorted ordering is on the key, and we also store a value in each node) and hash tables. One can implement a map using other data structures as well, such as a list.

A restriction on a map is that there is only one value associated with each key. Sometimes this is desirable, sometimes not. Consider the example of the courses that you are enrolled in this semester: the key is easy, as it's just your userid. But there are multiple values associated with that key, one for each class in which you are enrolled. We thus want to allow for a *set* of values for each key.

A *multimap* is a data structure that does just that: it allows for multiple values per key. Another way to explain it is that each key is associated with a *set* of values. Some operations on a multimap are:

- `insert(k,v)`: if the key k already exists in the multimap, then the value v is added to the (already existing) set of values for that key in the multimap; otherwise a singleton set (a set of just one element) containing just v is created as the set of value(s) for key k.
- `find(k,v)`: find if value v is in the set of values for key k.
- `remove(k,v)`: removes value v from the set of values for key k.
- `get(k)`: retrieves the (entire) set of values associated with key k; that set is returned as some data structure (array, list, iterator, tree – it doesn't really matter which one). This operation just obtains the set, it does not perform a search within that set.

When considering the size of the multimap, n refers to the *total* number of values in the multimap; the number of keys may very well be less than n.

Lastly, a multimap is a abstract data type, and can be implemented using a number of data structures. Which ones will be described in the questions below.

And in case you think this was made up for the exam, there is actually a `multimap` data structure in the C++ STL.

7. [4 points] Without knowing the underlying data structure that the multimap is implemented using, what is the most accurate estimate one can make for the running time of the four operations listed above?

Page 5: Multimaps, continued

8. [3 points] Multimaps can be implemented using a *balanced binary search tree* as the underlying data structure. Given such an implementation, what are the running times for the four operations listed above? Briefly, why?

9. [3 points] Multimaps can also be implemented using a *hash table* as the underlying data structure. Given such an implementation, what are the running times for the four operations listed above? Briefly, why?

10. [6 points] Describe how would you implement a multimap using a hash table. Keep in mind that your implementation must allow for key collisions, as well as (potentially) multiple values for each of the keys.

Page 6: Hash tables

[4 points] Some applications, such as real-time systems, cannot afford the time cost associated with rehashing the entire hash table when the load factor increases beyond a pre-set bound. Consider the concept of incremental resizing for hash tables. Instead of a full rehash, a second, larger, hash table will be created, and all new elements will be inserted into the second table; the original table is (initially, at least) left unchanged. Any search or delete will check both hash tables for the element. And after each insertion (into the second table), r elements (r is a constant) will be moved over from the original hash table to the second hash table. Eventually all of the elements will be moved over, at which point the original table will be deallocated.

11. [4 points] How does incremental hashing affect the running time of a hash table? If they are in the same big-Theta running class, you should still explain a differentiation between them, if one exists.

12. [4 points] To ensure that all the elements in the old table are copied over to the new table before the new table needs resizing, we can't increase the table size by too small an amount. By how much should we increase the table size to prevent this from occurring? Briefly, why?

Page 7: Heaps and Huffman Coding

13. [3 points] What is the difference between a priority queue and a (binary) heap?

14. [3 points] Although the worst-case running time for an insert into a binary heap is $\Theta(\log n)$, the typical running time is actually $\Theta(1)$. Why?

15. [3 points] In a min-heap, when percolating down, which child must you swap the parent node with? Why?

16. [3 points] What is the primary property of prefix codes that allows them to work in the context of Huffman coding?

Page 8: Graphs

17. [3 points] What is the running time of Dijkstra's algorithm? Why?

18. [3 points] Describe two ways that one can accelerate the Travelling Salesperson Problem (TSP). At least one must compute the exact correct answer (i.e., not an approximation algorithm); the other can be either an approximation or an exact result, but you must state which.

19. [3 points] Briefly describe the two MST algorithsm described in class: Prim's and Kruskal's. It's okay if you mix up the names; we're looking for knowledge of the algorithms.

20. [3 points] Describe an algorithm to determine if there is a cycle in a given graph. Note that you do not have to find the cycle itself, but just whether one exists. What is the algorithm's running time?

Page 9: Demographics

We meant to ask these in an end-of-the-semester survey, but we did not get to it in time. So we'll put it here for some extra points on the exam!

21. [0 points] Did you put your name and userid at the top of this page? You need to in order to get the points on this page.

22. [2 points] What is your major or minor (if you have not declared, then your intended major or minor)? Please circle one.

    - BS CS
    - BA CS
    - BS CpE
    - CS minor
    - Neither majoring nor minoring in computing
    - Other (please explain): _____

23. [2 points] What CS 1 class did you take? Please circle one.

    - CS 1110 (a.k.a. CS 101)
    - CS 1111 (a.k.a. CS 101-E)
    - CS 1112 (a.k.a. CS 101-X)
    - CS 1120 (a.k.a. CS 150)
    - AP credit
    - Transfer credit
    - Placed out of it via the CS 101/1110 placement exam
    - Other (please explain): _____

24. [2 points] If you took your CS 1 class in college (i.e. CS 101/1110, 101-E/1111, 101-X/1112, 150/1120, or a transfer class), in what semester did you take it?

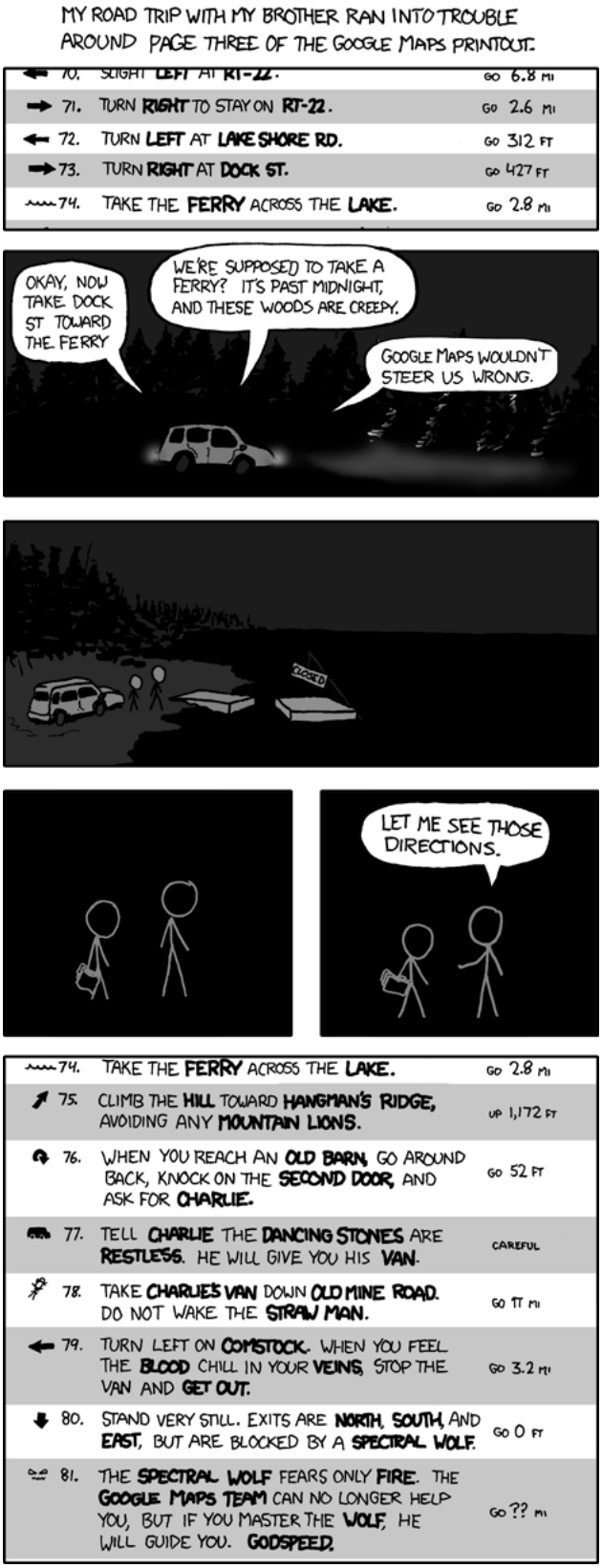
25. [2 points] What CS 2 class did you take? Please circle one.

    - CS 2110 (a.k.a. 201)
    - CS 2220 (a.k.a. 205)
    - AP credit
    - Transfer credit
    - Other (please explain): _____

26. [2 points] If you took your CS 2 class in college (i.e. CS 201/2110, 205/2220, or a transfer class), in what semester did you take it?

27. [2 points] Did you attend the final exam review session? You'll get full credit for this question, as long as you answer it honestly (we know most of the people that were there, but not all).

Page 10: When Dijkstra's shortest path algorithm fails...



(from `http://xkcd.com/461/`)