# CS 2150 final exam, fall 2015

**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!**

**Answers for the short-answer questions should not exceed about 20 words; if your answer is too long (say, more than 30 words), you will get a zero for that question!**

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.

*Three things are certain:*
*Death, taxes, and lost data.*
*Guess which has occurred.*

(the bubble footer is automatically inserted into this space)

## Page 2: Exam 1 and 2 stuffs

1. [6 points] Convert 0x00008dc1 (which is in little-Endian) into a floating point number using the IEEE 754 floating point notation. You can leave your result in formula form. Show your work!

2. [6 points] Give one advantage and one disadvantage of each of the collision resolution strategies that we have studied in this course. Note that you can't use the same reason twice! So if $A$ is faster than $B$, you can't *also* say that $B$ is slower than $A$.

| | Advantage | Disadvantage |
|---|---|---|
| Separate chaining | | |
| Linear probing | | |
| Quadratic probing | | |
| Double hashing | | |

## Page 3: C++

3. [3 points] Assume a `Student` class inherits from a `Person` class, and both define a `virtual print()` method. Given the code `Person *p = new Student();`, explain *all* the steps taken when `p->print();` is called (you don't need to get into the prologue here).

4. [3 points] Briefly describe the four types of multiple inheritance.

5. [3 points] Name three new features of C++11 (new relative to C++03, which is what we used this semester).

6. [3 points] When would we want to use non-`public` inheritance?

## Page 4: x86

7. [3 points] List the steps in the C calling convention's *caller prologue*.

8. [3 points] List the steps in the C calling convention's *callee prologue*.

9. [3 points] List the steps in the C calling convention's *callee epilogue*.

10. [3 points] List the steps in the C calling convention's *caller epilogue*.

## Page 5: Heaps and Huffman

11. [9 points] Consider a priority queue implemented with various data structures. Fill in the table below with their big-Theta run times.

| | `insert()` | `findMin()` | `deleteMin()` |
|---|---|---|---|
| Unsorted vector | | | |
| Unsorted linked list | | | |
| Sorted vector | | | |
| Sorted linked list | | | |
| Binary search tree | | | |
| AVL or red-black tree | | | |
| Hash table | | | |
| Binary heap | | | |

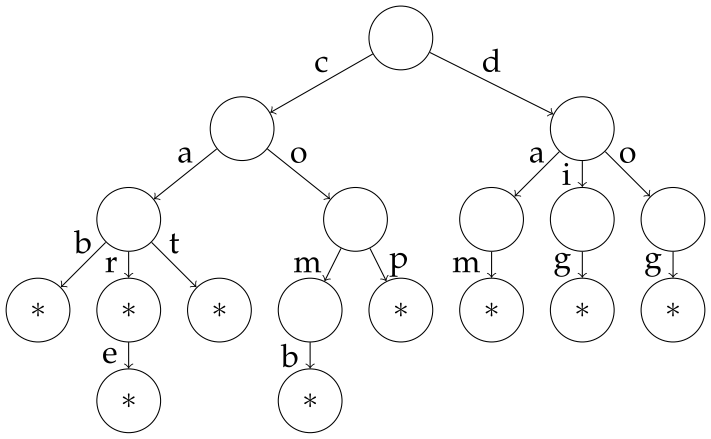
12. [3 points] Briefly describe the heap sort algorithm.

## Page 6: Tries

A trie (typically pronounced like the English word "try") is a particular kind of tree (NOT necessarily binary) associated with an alphabet where each node can have up to $d$ children where $d$ is the number of letters in the alphabet. For example, if the alphabet we are using with a trie is the set of uppercase English letters (that is, 'A' to 'Z'), then each node can have up to 26 children.

Typically, tries are used to store sets of strings for fast look-up by prefix. Strings of letters from the alphabet are usually encoded by configuring the nodes of the tree in the following way: (1) start at the root; (2) look at the first letter of the string, and move down to the child of the root associated with that letter (allocating this node if necessary); (3) for each successive letter, move from whatever node we are currently at to the child of that node associated with that letter (again, allocating memory for it if necessary); and (4) when at the end of the string, mark the final node as the end of the word by setting a Boolean that we store in each node. This final step allows us to accommodate words in our dictionary that are prefixes of other words. Note that every leaf node will mark the end of a string, and if no strings are prefixes of other strings, then *every* word ends on a leaf node.

For example, suppose that we wanted to place the following dictionary into a trie: {cab, car, care, cat, comb, cop, dam, dig, dog}. Then, our trie would look like this:



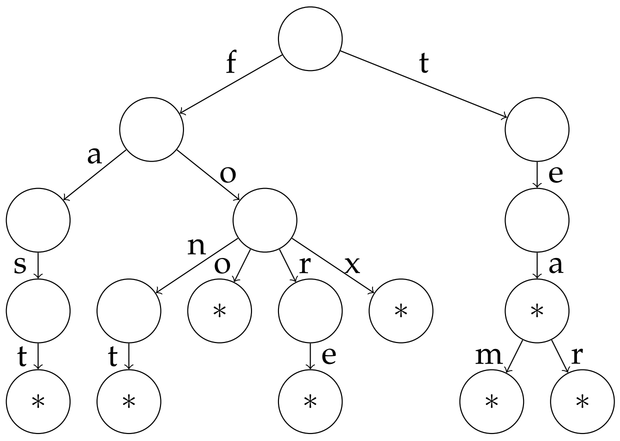
In the above diagram, nodes that are the end of some word in the tree are marked with an asterisk ($*$) while all other nodes are left blank. In particular, all leaves and one interior node (the node reachable by the string "car") mark the ends of words and thus are marked as so, with an asterisk, in the tree.

Now, if one wanted to check to see if the string "cast" is in the trie, one can do so by noting that there is an edge from the root to one of its children through edge 'c', and an edge 'a' from that node to another, but there is no child of *that* node associated with the letter 's', so we can determine that "cast" is *not* in the trie. Likewise, we can tell that "dig" *is* in the trie because we can follow edge 'd' from the root, then edge 'i', and then the edge 'g' to a node marked as the end of a word (which happens to be a leaf). Although "do" leads to a valid node, it is not marked as a final node (via an asterisk), so the word "do" is not in the dictionary.

Note that for the following questions, you can assume that it is a $\Theta(1)$ operation to determine if there is an edge from one node to one of its children associated with some letter in the alphabet and also a $\Theta(1)$ operation to follow the edge if such an edge exists. (In practice, this is usually done using an array of pointers to children where the position in the array corresponds to the appropriate letter in the alphabet.)

## Page 7: Trie Questions, page 1

13. [3 points] Consider the following trie (using the same format as the previous page). List all of the words included in the trie.



14. [6 points] Construct a trie out of the following words (using the format of the trie above and the example): {gem, gems, gist, gone, so, sock, son, sore, sour, sum}.

15. [3 points] Suppose you have a trie constructed out of $n$ words, each one containing at most $m$ letters. Now suppose you are given some word. What is the big-Theta of checking membership of this word in the trie in terms of $n$ and $m$?

## Page 8: Trie Questions, page 2

16. [6 points] Think back to lab 6 (hashes). Suppose you have a dictionary with $n$ words in it and each word has at most $m$ letters. Now, suppose you have constructed a trie out of these $n$ words, and you are given a grid with $r$ rows and $c$ columns that you have read into a two-dimensional array. Recall that each word can appear in one of eight directions. Briefly describe what algorithm you could run using your trie to solve the word search.

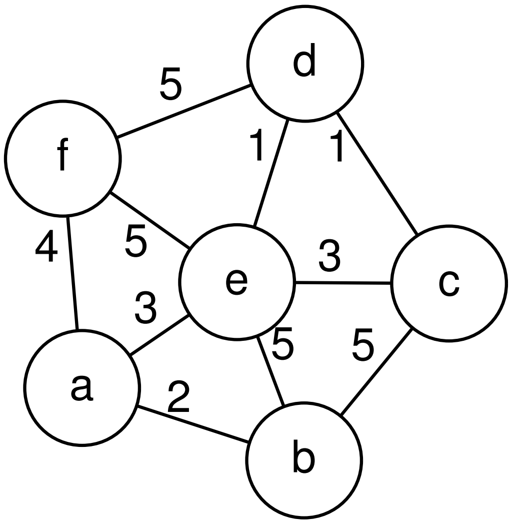
17. [3 points] What is the running time of your algorithm from the the previous question? Please clearly state the big-Theta trie lookup time is (this is from two questions back) in addition to the overall algorithm run time of the algorithm from the previous question. This big-Theta run time should be in terms of $r$, $c$, $n$, and $m$.

18. [3 points] Give two examples of when a trie is preferable over a hash table. Also, give two examples of when a hash table is preferable over a trie. Note that you can't use the same reason twice! So if *A* is faster than *B*, you can't *also* say that *B* is slower than *A*.

## Page 9: Graphs and Memory

19. [6 points] Given the following graph, perform Dijkstra's shortest path in the table below. If a cell is updated, be sure to include both the original value(s) (crossed out), as well as the updated value(s). Start at node "a".

| node | known? | distance | path |
|---|---|---|---|
| a |  | 0 |  |
| b |  |  |  |
| c |  |  |  |
| d |  |  |  |
| e |  |  |  |
| f |  |  |  |



20. [3 points] Using one of the algorithms taught in lecture, how would you find if a cycle exists in a graph? You don't have to find the exact cycle; just determine if one exists.

21. [3 points] Consider a list data structure, which can be implemented with arrays (likely vectors) or as a linked list (with dynamically allocated list nodes). For this question, we will ignore the cost of expanding a full vector, which means that the operations we care about have the same running time with both implementations (those operations: push_back(), pop_back(), and iteration through the list). Give three memory-related reasons why the array implementation would be better (either faster speed or take up less memory) than the linked list implementation.

## Page 10: Demographics Name & userid: ______________________________

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! Sorry if this page is a bit crowded...

22. [0 points] Did you put your name and userid at the top of this page? You need to do so in order to get the points on this page!

23. [2 points] What is your major or minor? If you have not declared, then answer with your intended major or minor. Please circle one.

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

24. [1 points] Have you already declared the major/minor mentioned above? Circle: Yes or No

25. [2 points] What CS 1 class did you take? Please circle one.

    - CS 1110
    - CS 1111
    - CS 1112
    - CS 1120
    - AP credit
    - Transfer credit
    - Other (please explain): ______________________
    - Placed out of it via the CS 1110 placement exam

26. [1 points] If you took your CS 1 class in college (i.e. CS 1110, CS 1111, CS 1112, CS 1120, or a transfer class), in what semester did you take it? Please specify a semester by season and calendar year (i.e., "fall 2012" and not "my second year").

27. [2 points] What CS 2 class did you take? Please circle one.

    - CS 2110
    - CS 2220
    - AP credit
    - Other (please explain): ______________________
    - Transfer credit
    - Placement exam

28. [1 points] If you took your CS 2 class in college (i.e. CS 2110, CS 2220, or a transfer class), in what semester did you take it? Please specify a semester by season and calendar year (i.e., "fall 2012" and not "my second year").

29. [1 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 that were there, but not all).

30. [2 points] For the 3-credit courses for next semester (not summer or J-term):

    - How many CS courses are you enrolled in (not wait-listed)?
    - How many CS courses are you wait-listed for?
    - How many CS courses would you *like* to be enrolled in?