CS 2150 Final Exam

You MUST write your name and e-mail ID on EACH page and bubble in your userid at the bottom of EACH page – including this page.

If you are still writing when "pens down" is called, I will rip up your exam and give you a zero.

There are 10 pages to this exam – once the exam starts, please make sure you have all the pages. Pages 2-10 are worth 12 points each, for a total of 108 points. The last page will take significantly less time than the other pages. The first page is worth 5 points, to encourage you to bubble in that page. Thus, the entire exam is worth 113 points – and, by the way, that number is prime, and would thus make an excellent hash table size. With 180 minutes (3 hours) for the exam, and 96 points (not counting the first and last pages), you should spend just under 2 minutes per question point. The long section of reading (page 4) is worth free points to give you time to read those pages.

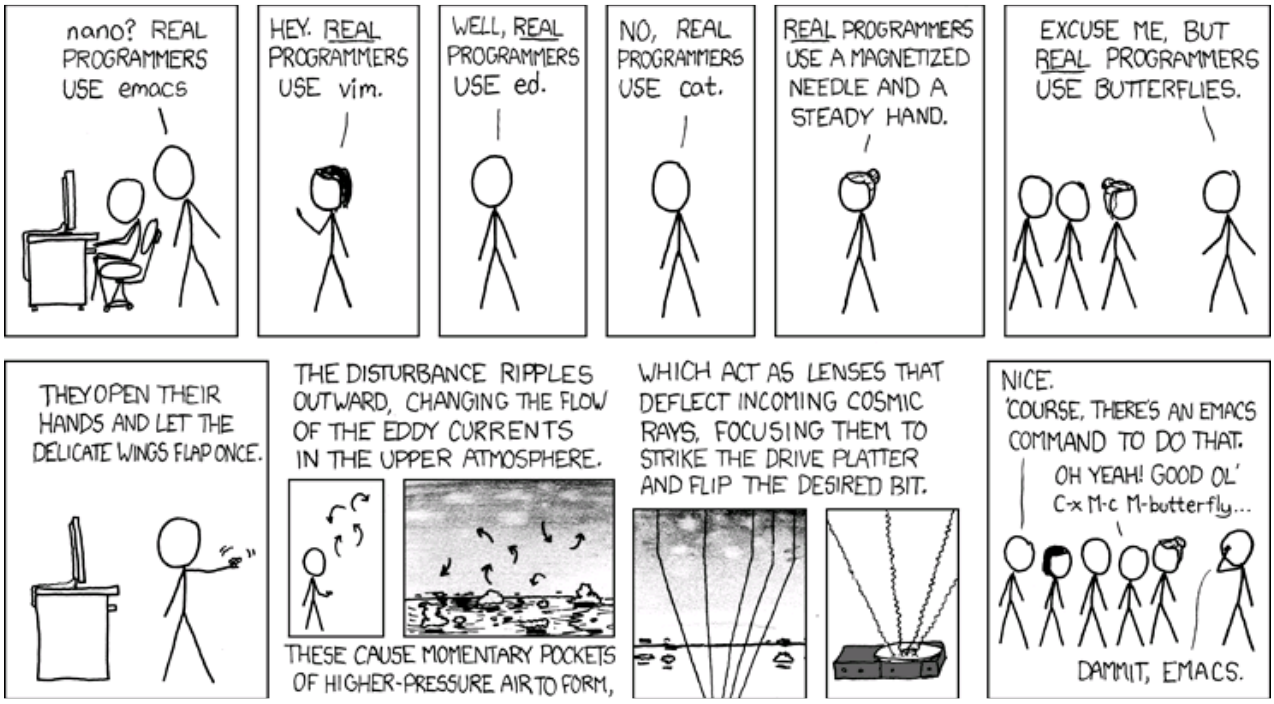
If you do not bubble in a page, you will not receive credit for that page!

This exam is CLOSED text book, closed-notes, **closed-calculator**, closed-cell phone, closed-computer, closed-neighbor, etc. Please sign the honor pledge here:

_____

_____

_____



(the bubble footer is automatically inserted in this space)

C++

1. [6 points] Write a small C/C++ code snippet that creates a buffer overflow attack. It doesn't have to do anything useful – just overflow a buffer in the same manner as the type of attacks we talked about in lecture.

2. [3 points] Briefly, what are virtual method tables? Why are they needed? How do they work?

3. [3 points] What are the 4 types of multiple inheritance, and (briefly!) what does each one mean?

Numbers

4. [6 points] Convert -0.05078125 (which, to save you time, is equal to -13/256) to **little**-endian IEEE 754 single-precision floating point number format. Your answer should be in hexadecimal.

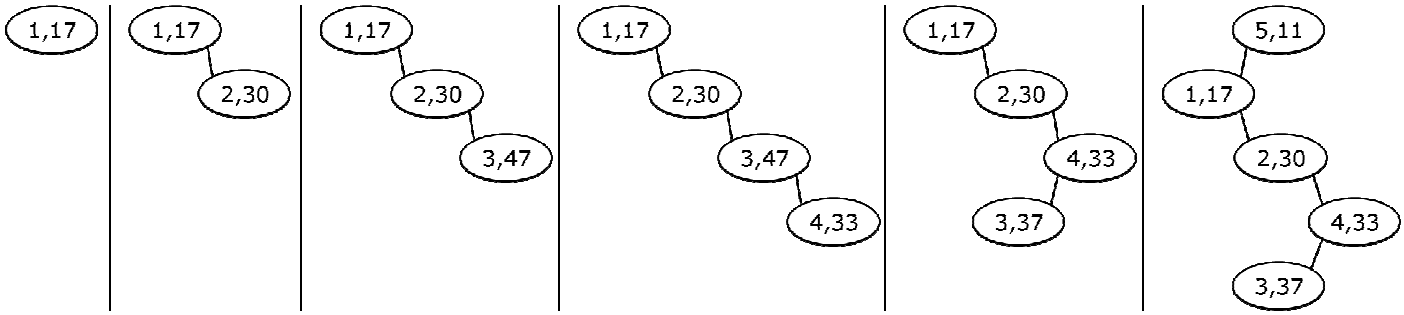
5. [6 points] Encode -170 as a 32-bit little Endian two's complement integer (note the negative sign on that number). Show you work for partial credit!

A *treap* is a data structure that combines the properties of both trees and heaps, hence it's name ("treap" = "tree heap"). A treap node has two children (thus allowing the binary tree structure), a value (the data that we are storing; we'll assume integers for this exam), and a *priority*. The value is sorted as per the binary search tree ordering property, while the priority is sorted as per the min-heap ordering property. Thus, a valid treap must fulfill both of these properties – although it is not obvious that both constraints can be simultaneously satisfied, it can be done, as explained below.

Given a series of values to insert – we'll assume the numbers 1, 2, 3, 4, and 5 – the treap assigns to each node a *random* priority. We'll assume the priorities assigned to them are, respectively, 17, 30, 25, 33, and 11. In reality, the priorities range across the full range of integers.

A treap insertion begins like a normal BST insertion – the node is inserted into the unique position in the tree that fulfils the BST ordering property – recall that the BST ordering property for the treap is based on the value of each node, not the priority. Next, the inserted node is rotated so that it's priority is properly positioned as per the heap ordering property. Recall that tree rotations always preserve the BST ordering property, so we will use them to fix the min-heap ordering property of the priorities, as described more below. Also recall that a rotation moves at least one node up in the tree, and one node down.

Consider inserting the five elements described above (the integers 1-5, and their respective priorities). We insert them in the order specified: (1,17), (2,30), (3,47), (4,33), (5,11). Note that the first number in the ordered pair is the value, and the second is the priority. The first three insertions are easy, as the values and priorities are (intentionally) properly positioned as per their respective ordering properties without any necessary tree rotations. These are the first three images below.



Insertion of the next value, (4,33), inserts to the bottom of the tree as per the BST ordering property (4th image, above). But this causes the priorities to not fulfill the min-heap ordering property. Thus, the bottom two nodes are rotated – this preserves the BST ordering property and allows us to achieve the min-heap ordering property of the priorities. This is the 5th image above.

The last value to be inserted, (5,11), is inserted as the right sub-tree of (4,33) (this is not shown in the images above). However, in order to preserve the min-heap ordering property on the priorities, we must rotate that node repeatedly until it becomes the new root node, since it has the lowest priority – the final resulting treap is shown as the right-most image above. Note that the unbalanced-ness of the final treap is an artifact of the particular values we chose to use in this example.

Find is identical to the BST find. Remove is similar to insert – perform a BST remove, then rotate to ensure that the new nodes' priorities fulfill the min-heap ordering property.

Treaps

6. [3 points] Why would we want to use a treap over a BST? Hint: one reason has something to do with the randomization of the treap.

7. [3 points] What is the running time of the primary operations (insert, find, remove)? Is this an accurate representation of the performance of the treap? If not, give an alternate running time (and explanation), such as the expected running time.

8. [3 points] What are the advantages of a treap over BSTs? Over AVL trees?

9. [3 points] What are the disadvantages of a treap over BSTs? Over AVL trees?

Trees & Hashes

10. [6 points] Consider the three types of self-balancing trees that we have seen: AVL, red-black, and splay. BRIEFLY give one advantage and one disadvantage of each of these trees. If it's not obvious, explain why.

11. [6 points] Consider the four types of collision resolution that we have seen: separate chaining, linear probing, quadratic probing, and double hashing. BRIEFLY give one advantage and one disadvantage of each of these types of collision resolution.

Miscellaneous

12. [6 points] List all the steps of the x86 calling convention – you need to list both prologues and both epilogues.

13. [3 points] List three traveling salesperson acceleration techniques that can be used to avoid the worst-case running time.

14. [3 points] What two properties of computer memory access allow cache memory to be effective? Briefly, what do those two properties mean?

Other Programming Languages

15. [3 points] Name one distinctive feature of Objective C, as compared to C++.

16. [3 points] If you had to explain what aspect-oriented programming was to a fellow computing major in 20 words or less, how would you do so?

17. [3 points] Name one distinctive feature of Intercal.

18. [3 points] If you had to add one feature to IBCM, balancing both feasibility of adding that feature to the language and necessity of having said feature in the language, what feature would you add? Why?

UNIX

19. [3 points] Define the following terms from Makefiles (and briefly describe what they do and/or what they are used for): suffix rules, dependencies, targets.

20. [3 points] List three doxygen tags that you know, and give a (brief!) explanation of each.

21. [3 points] List all the Emacs keyboard commands that you know, and give a BRIEF description of what each one does (a word or two is fine here). At this point in the semester, you should know at least 8. And C-x M-c M-butterfly doesn't count… ☹

22. [3 points] List 2 reasons why the hash table size must be prime (I can't figure out the Unix connection here, either. But I was down a Unix question, and I wanted to ask this one. So here you go).

Demographics

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

23. [0 points] Did you put your name at the top of this page? You need to in order to get the points on this page.

24. [2 points] What is your major or minor (if you have not declared, then your intended major or minor)?

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

_____

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

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

_____

26. [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?

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

- CS 201 (now 2110)
- CS 205 (now 2220)
- AP credit
- Transfer credit
- Other (please explain):

_____

28. [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?

29. [2 points] Did you attend the 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).

(the bubble footer is automatically inserted in this space)