CS 2150 Exam 2

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. The first and last pages are not worth any points; the other pages are worth 12 points each, for a total of 96 points on this exam.

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.

_____

_____

_____

_____

_____

First snow, then silence.
This thousand dollar screen dies
So beautifully.

(the bubble footer is automatically inserted into this space)

Page 2: Exam 1 material

1. [3 points] When is a destructor called? List two different things that causes a destructor to be called.

2. [3 points] Give two differences between how C++ treats an array base name versus a pointer.

3. [3 points] Consider a `short`, a signed 8-bit two's complement integer type. What is the minimum and maximum value that this type can hold?

4. [3 points] What is the difference between a shallow copy and a deep copy?

Page 3: Trees

5. [3 points] Describe the full algorithm for removal of a binary search tree node with two children

6. [3 points] In an AVL tree, we only need to perform at most two rotations (i.e. a double rotation) to fix the tree. Why don't we need to perform more rotations?

7. [6 points] What are the 5 properties of red-black trees?

Page 4: Trees, page 2

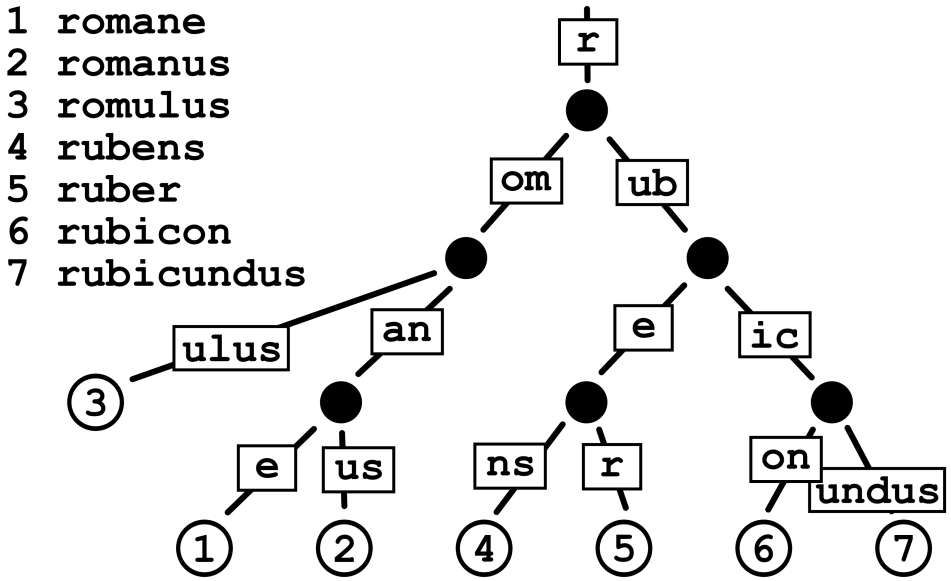
8. [6 points] Consider the three balanced binary trees that we have studied: AVL, splay, and red-black. Give one advantage and one disadvantage of each.

9. [9 points] Draw an AVL tree that is unbalanced, and in need of a double rotation (either type) for proper re-balancing. Show the balance factors for the nodes, as well as the values in the nodes (they can be any values, but use integers). Show the tree after the first part of the double rotation (with node values and balance factors), as well as when completed (i.e. after the second part of the double rotation).

Page 5: String Trees

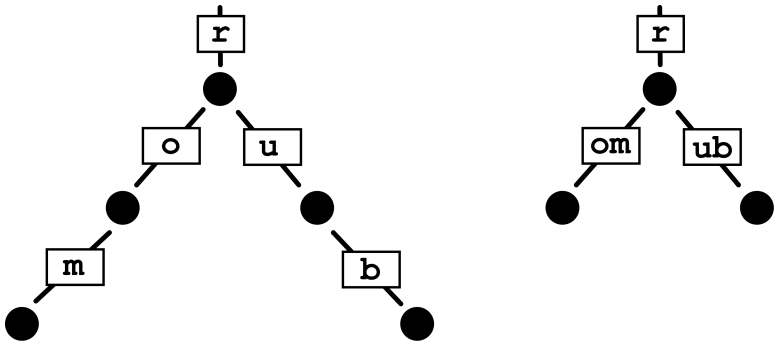
While this may seem like a lot to read, you get 12 free points for reading this page.

Consider a new data structure called a *string tree*. A string tree contains a string label on each edge. If you follow the path from the root node to a leaf node, you concatenate the strings along the edges of the path to form a word stored in the tree. In the diagram shown to the right, if you follow the path from the root node *r* to leaf node 3, you will get the word "romulus". Each leaf node, then, represents a single word, and can be used to contain additional information, such as the definition. In this diagram, the seven words that the string tree holds are listed by number in the upper left (the numbers here are arbitrary, and are just for illustration).



A string tree can have many children per node. If we inserted the word "right" into the tree shown in the above diagram, then the root node *r* would have three children; if we also inserted "rest", then the root node *r* would have four children.

String trees also have the property that a node with only one child is merged with that child, and the edges are updated appropriately. Consider the first string tree to the right, which is a part of the string tree shown above. The node that is the left child of the root node is combined with it's child, and the edges are concatenated together, so that the 'o' edge and the 'm' edge are combined to form a single "om" edge. Similarly with the right sub-tree, where 'u' and 'b' are combined to form "ub". The result would be the second string tree shown in that diagram. Note that the rest of the tree was removed for clarity in this diagram.



The next few questions are about string trees.

Page 6: String Trees, page 2

10. [4 points] The primary operations in a string tree are insert, remove, and find. What are their running times? Explain your answer.

11. [4 points] As described on the previous page, only leaf nodes can signify words. Consider if we wanted to allow the words "roman" or "rub", both of which are in the tree, but terminate at internal nodes. How would we modify our string tree to allow for including such words?

12. [3 points] Contrast a string tree to a balanced binary tree, such as an AVL tree. List one advantage and one disadvantage of a string tree versus an AVL tree.

13. [3 points] Contrast a string tree to a hash table. List one advantage and one disadvantage of a string tree versus a hash table.

Page 7: Hashes

14. [3 points] What are the three necessary properties of a good hash function?

15. [3 points] What was the in-class exercise that was used to explain hash functions?

16. [3 points] When using an open addressing (a.k.a. probing) collision resolution strategy, removing an element can cause 'issues'. List three ways that a hash table can manage these 'issues' in the remove operation.

17. [3 points] What is thrashing? When does it occur? How do we solve it?

Page 8: IBCM

18. [3 points] One can implement a stack in IBCM analogous to how the x86 stack works: it starts at the end of memory (address 0xfff), and moves towards the beginning of memory on each push.

    You need to write the push code for the push operation. This is analogous to the `push` opcode in x86. Assume that the stack pointer is stored in address 0x001, which you can refer to by the label 'sp'. Your program is not intended to be a complete IBCM program or a proper subroutine – just the necessary code to perform a push operation.

    The IBCM opcodes (and their hex values) are shown to the right for your reference. Your IBCM program should be written in opcode form, and **NOT** written in hexadecimal. If you leave your code in hexadecimal machine language, we will not grade it, and you will not receive credit.

Hex	Opcode
0	halt
1	I/O
2	shifts
3	load
4	store
5	add
6	sub
7	and
8	or
9	xor
a	not
b	nop
c	jmp
d	jmpe
e	jmpl
f	brl

Page 9: x86

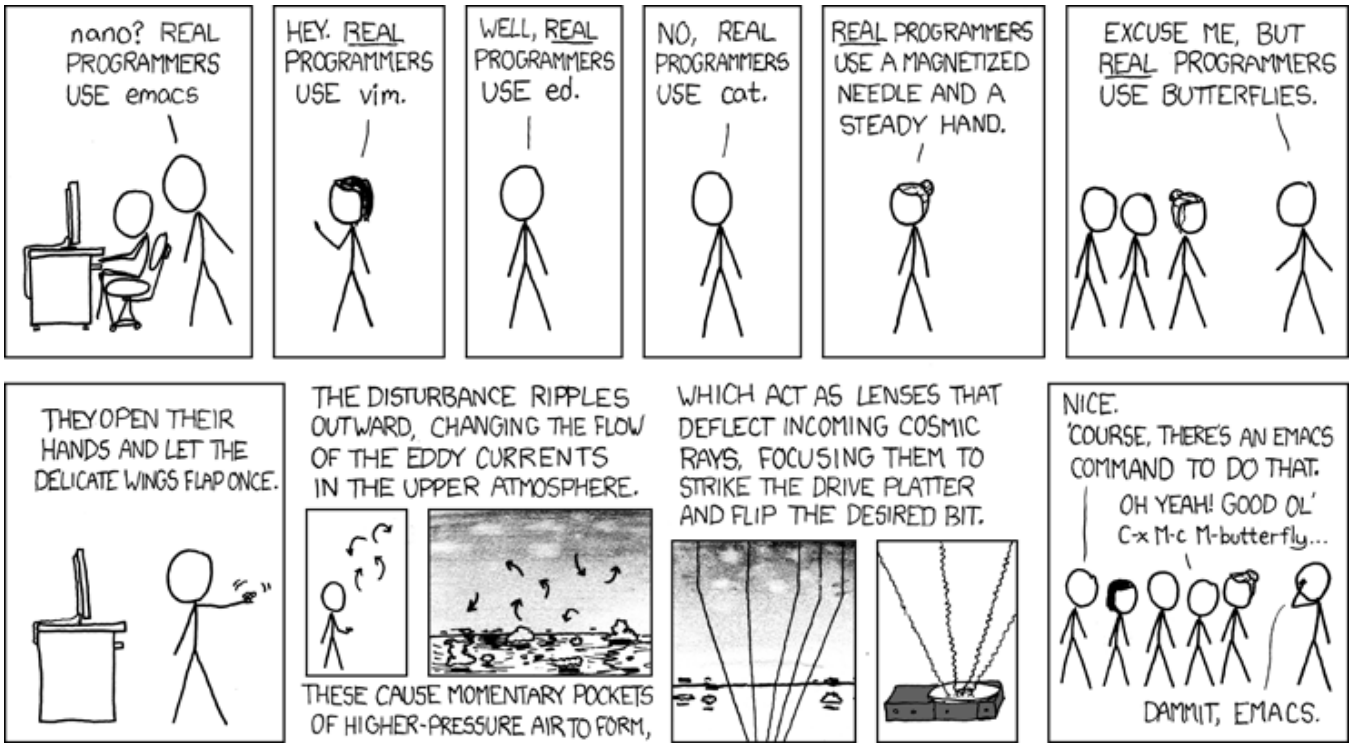
19. [3 points] What are the steps necessary in the x86 C calling convention caller's prologue?

20. [3 points] What are the steps necessary in the x86 C calling convention caller's epilogue?

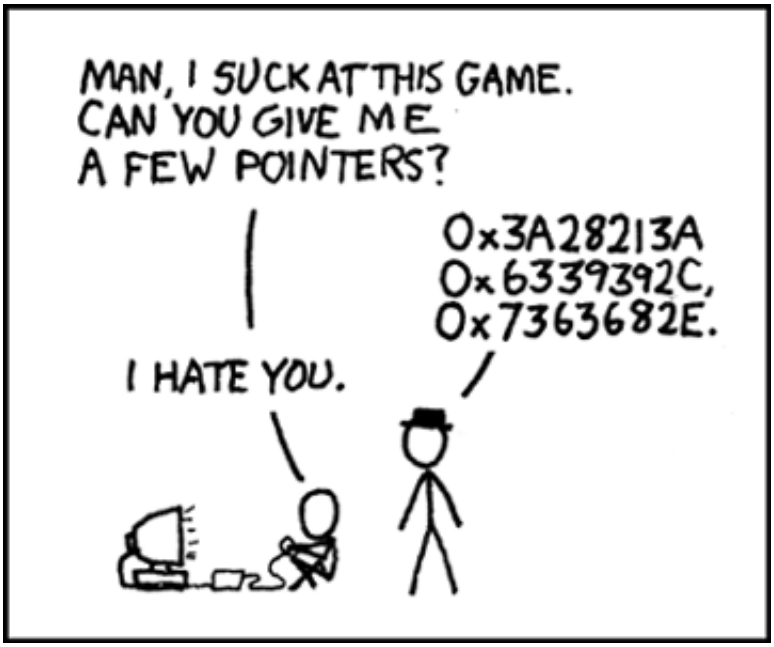
21. [3 points] What are the steps necessary in the x86 C calling convention callee's prologue?

22. [3 points] What are the steps necessary in the x86 C calling convention callee's epilogue?

Page 10: Comics!



(from `http://xkcd.com/378/`)



(from `http://xkcd.com/138/`)