CS 2150, fall 2015
Final exam for tm5gf
Tushar Maharishi

Your original score on this exam was 77.5 of 96 points (80.73%).

Your breakdown of points per page is below.

Page	Score	Max
1	0	0
2	11	12
3	6.5	12
4	8	12
5	7.5	12
6	0	0
7	12	12
8	10	12
9	10.5	12
10	12	12

A graded scan of each page follows.

CS 2150 final exam, fall 2015

Name Tushar Maharishi (tmsgf)

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!

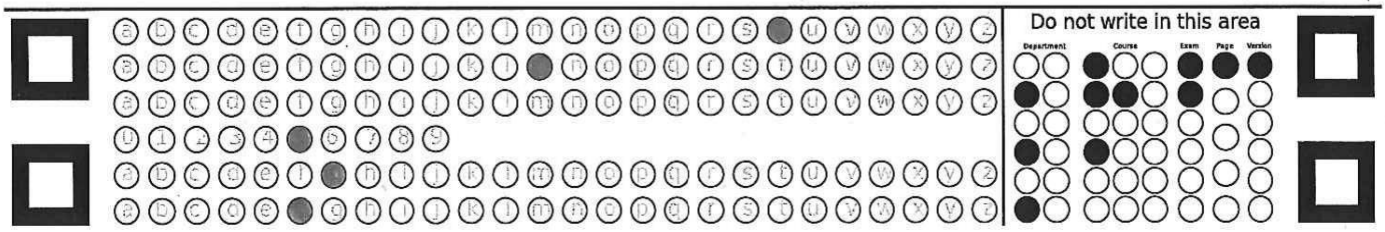
Answers for the short-answer questions should not exceed about 20 words; if your answer is too long (say, more than 30 words), you will get a zero for that question!

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.

I pledge on my honor that I have neither given nor received aid on this exam.

Tushar Maharishi

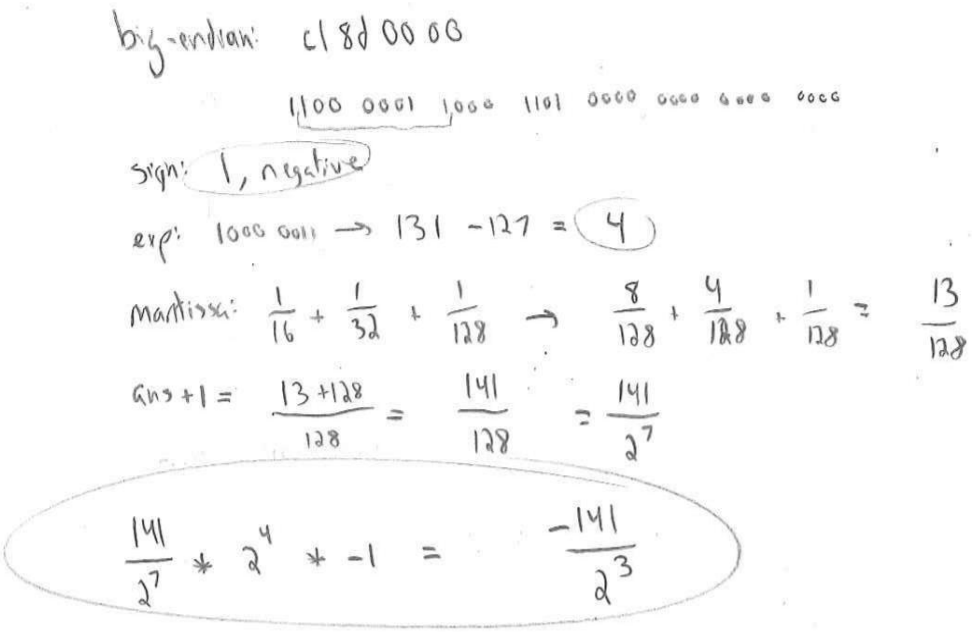
Three things are certain:
Death, taxes, and lost data.
Guess which has occurred.



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!

6/6



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*.

5/6

	Advantage	Disadvantage
Separate chaining	easy to perform deletes	~~lots of overhead~~, uses additional memory to create buckets
Linear probing	easy to implement	~~tends to create clusters in the table~~ can create "holes" when deleting
Quadratic probing	avoids clustering	requires slightly more computation power than linear probing
Double hashing	very fast run-time	double hashing thrashing! (can be fixed with prime table size)

The score on this page is 11/12

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).

1/3

The constructor for person is called. Then a virtual method table is created and the print() method is stored there. Once the constructor for Student is called, its print() method is added to update the table. Then p->print(); calls the Student's print().

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

2.5/3

shared : uses virtual method table to distinguish method being used
replicated : default for C++, possibly for two inherited classes to have different definitions for same method name.
non-replicated : does not allow any replication in inheritance
mix-in : what JAVA uses, "fakes" multiple inheritance using interfaces

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

3/3

- introduces for/range based loops.
- has uniform instantiation
- creates a specific nullptr for pointers

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

0/3

~~when we want to hide~~

when the class that inherits may not be able to sensibly implement all of the parent classes functions. Private and protected are there to limit visibility so only some functions can be inherited.
there is an option that

The score on this page is 6.5/12

Page 4: x86

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

2.5/3

push ~~registers~~ caller-saved registers

push parameters in reverse order

call subroutine

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

2/3

pop parameters

return value stored in EAX

restore caller-saved registers

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

1.5/3

push ebp and move esp into ebp

decrement esp why?

push callee-saved registers

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

2/3

store return value in EAX

deallocate local variables

restore callee-saved registers

pop ebp

return

The score on this page is 8/12

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.

5/9

	insert()	findMin()	deleteMin()
Unsorted vector	1	n	n
Unsorted linked list	1	n	n
Sorted vector	n	1	1
Sorted linked list	n	1	1
Binary search tree	log(n) X	log(n) X	log(n) X
AVL or red-black tree	log(n)	log(n)	log(n)
Hash table	1 X	n	n
Binary heap	1 (amortized)	1	log(n)

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

2.5/3

keep all the elements in a heap format and run deleteMin on the heap until no elements remain. Thus the smallest elements are popped off first and will be stored in sorted order.

Has $n \log(n)$ run-time but is less reliable than other sorts like mergesort.

The score on this page is 7.5/12

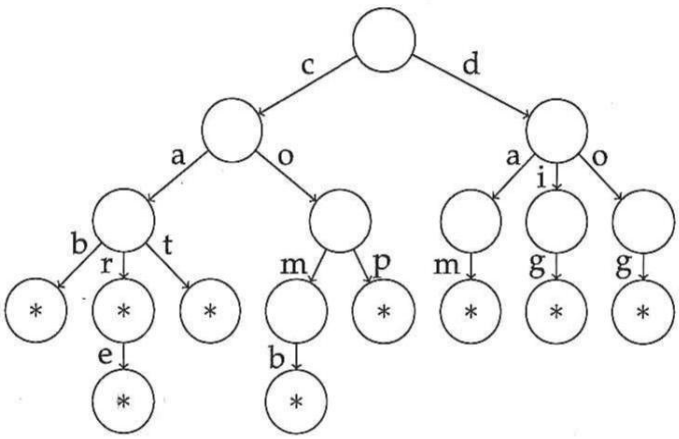
would still insert into the heap first

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.

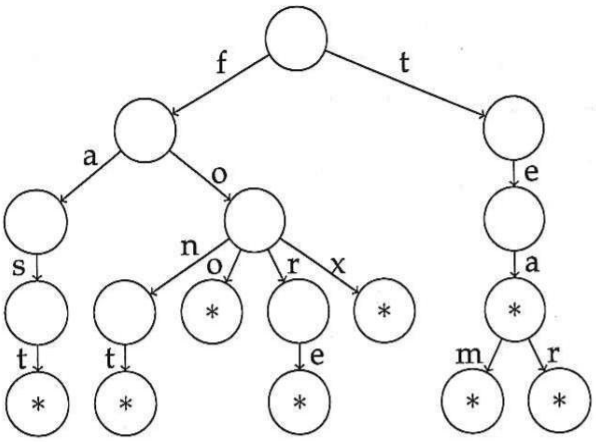
Now, if one wanted to check to see if the string "cast" is in the trie, one can do so by noting that there is an edge from the root to one of its children through edge 'c', and an edge 'a' from that node to another, but there is no child of *that* node associated with the letter 's', so we can determine 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.)

↑ smart!

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.

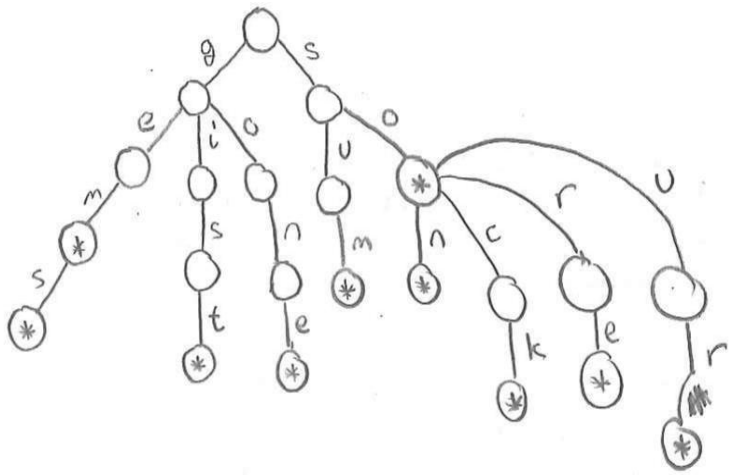


fast tear
fart
foo
fore
fox
tea
team

3/3

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, stum}.

6/6



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?

3/3

~~$\Theta(n+m)$~~ $\Theta(m)$ because the number of words is irrelevant as determining edge presence is $O(1)$. Thus, only word size matters. For an m-sized word, it will be m levels deep.

The score on this page is 12/12

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.

instead of checking the hash table for presence of every combination in each direction, check if the current letter combination exists in the dictionary trie. If yes, then continue searching in that direction for future words. If no, then stop and move on to the next direction or ~~position~~ index of the matrix.

6/6

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.

$\Theta(r * c * m)$

for each row and column, the algorithm needs to check, at worst case, the number of levels of the trie (which is m) 8 times. 8 is constant and can be ignored.

$\Theta(m)$ is lookup from question 15.

3/3

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.

Trie: 1) better when looking up similar patterns with small changes as prefixes
~~2)~~ could also be used in checking presence/legitimacy of permutations of words

Hash: ~~1)~~ better when looking up more random combinations of letters
2) better when elements have no structuring property (no order of letters in words) but find and insert are still necessary functions.

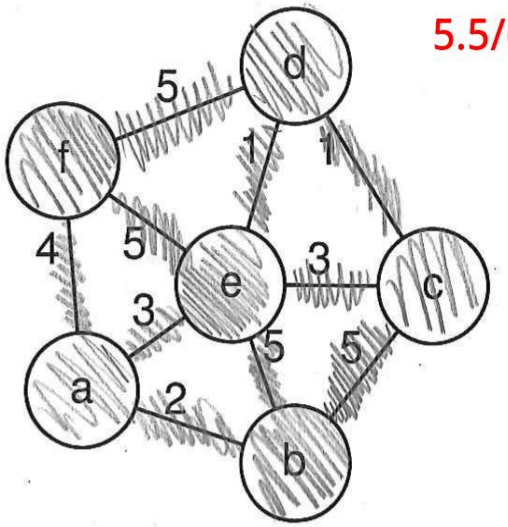
1/3

The score on this page is 10/12

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	✓	2	a
c	✓	~~7~~ ~~6~~ 5	~~b~~ ~~e~~ d
d	✓	4	e
e	✓	3	a
f	✓	4	f ✗



5.5/6

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.

insert a node into a queue and mark it as known.
pop off queue and add all non-known neighbors into queue and mark as known.
repeat until a known node is found (cycle exists!) or no more nodes. (no cycle!)

3/3

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.

- better for caches (arrays are stored together in memory)
- no need to waste memory storing pointers to next/previous nodes
- constant look-up time ✗

2/3

The score on this page is 10.5/12

Page 10: Demographics

Name & userid: tushar Maharishi

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"). n/a, AP credit

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"). fall 2014

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). yes!

30. [2 points] For the 3-credit courses for next semester (not summer or J-term): 6/6

- How many CS courses are you enrolled in (not wait-listed)? 2
- How many CS courses are you wait-listed for? 5 6/6
- How many CS courses would you *like* to be enrolled in? 3

The score on this page is 12/12