CS 2150 Exam 2, fall 2019

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 6 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.

_____

_____

_____

_____

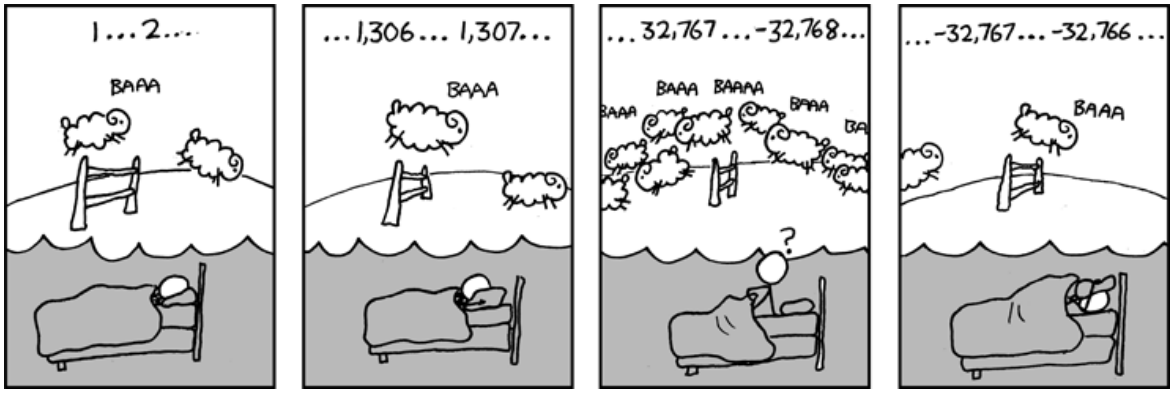
You step in the stream,
But the water has moved on.
This page is not here.

(the bubble footer is automatically inserted into this space)

Page 2: Exam 1 material

1. [6 points] Suppose you are writing a queue with `front` and `back` indices, as well as an array in an `array` field. Write out the `enqueue(...)` method. You may assume the method `resize(count)` exists which resizes the array to the given count while also updating front and back as appropriate. Additionally, `size()` returns the current number of elements in the array, while `capacity()` returns the maximum number of elements that can fit in the array without resizing. Solutions will be graded both on correctness and efficiency.

2. [3 points] Explain the joke / punchline in the comic below (from xkcd #571)



3. [3 points] For floating point numbers, what is the trade-off between having more bits in the mantissa versus more bits in the exponent?

Page 3: Trees

4. [5 points] Consider a method `getCommonAncestor()` that will find the *lowest* common ancestor between two AVL tree nodes; the function is also supplied the root node of the AVL tree. These AVL nodes do *not* have parent pointers. This function must run in $\Theta(\log n)$ time. Give an *English* description of how this method would work.

5. [5 points] Write the *recursive* method `int sumLeaves(BSTNode *node)` that sums the values in the leaf nodes (and only in the leaf nodes!) for the passed tree. Assume each `BSTNode` has a `left` and `right` child pointer, as well as a `value` field.

6. [2 points] Give one reason why red-black trees are faster than AVL trees

Page 4: Hashes

7. [3 points] Give two reasons why a hash table size should be prime.

8. [3 points] For the hash function $hash(k) = (k + 2)\%tableSize$, state which properties of a good hash function the function does and does not meet.

9. [3 points] Consider the hash table below. The hash function is simply $hash(k) = k\%tableSize$. For *each* of the probing strategies, provide a value that will have at least *three* collisions before finding the right spot in the table. For double hashing, you'll have to provide the secondary hash function.

index	key
0	60
1	
2	
3	23
4	
5	55
6	36
7	97
8	
9	19

Page 5: IBCM and Assembly

10. [6 points] Consider the following *buggy* IBCM program that computes a product by iteratively adding numbers together, and stores the result in the first number. Briefly describe the bug(s) in this code AND fix the code on the left to correct the bug(s). You can assume that in addition to x and y, you have the variables `zero` and `temp`, both initialized to 0.

```
// multiply x and y, storing
// the result in x
mult:
   load x
   store temp
loop:
   load y
   jz done
   load x
   add temp
   store x
   load y
   sub 1
   store temp
done:
   halt
```

11. [6 points] Below is a function to calculate the n^{th} Fibonacci number (the Fibonacci series is a series of numbers in which each number is the sum of the two preceding numbers, and starts as: 1, 1, 2, 3, 5, 8, ...). It's missing three lines of code – fill them in.

```
fib:
   xor rax, rax
   cmp rdi, 0
   je done
   cmp rdi, 2
   jle base
   push rax
   dec rdi

   ---------------
   call fib

   ---------------

   ---------------
   mov r10, rax
   push rax
   dec rdi
   call fib
   pop r10
   sum rax, r10
   ret
base:
   mov rax, 1
done:
   ret
```

Page 6: Miscellaneous

12. [3 points] Give a realistic example of how one would use big-Omega.

13. [3 points] Briefly explain the purpose of the `chmod` Unix command and give one example of where it might be used.

14. [3 points] Briefly describe what both targets and dependencies are in Makefiles.

15. [3 points] Briefly explain why the insert operation in a vector is $\Theta(1)$ amortized.