CS 2150 Exam 2, fall 2016

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: Old stuff

1. [3 points] What is the output of this program? It does successfully compile and run!

```
#include <iostream>
using namespace std;
class C {
 public:
  C() {}
  C(const C&) { std::cout << "A copy was made." << endl; }
};
C f() {
  return C();
}
int main() {
  cout << "Hello World!" << endl;
  C obj = f();
  return 0;
}
```

2. [3 points] What is the largest `double` value that can be represented? Explain your reasoning! You are welcome to leave your answer as an expression (i.e., equation).

3. [3 points] Consider an 8-bit two's-complement *signed* integer type (just like `byte` in Java); we'll call that type `byte` here as well. Given the code `byte a = 100, b = 50, c = a+b;`, what value is in `c`?

Page 3: Trees

4. [3 points] Briefly list two reasons you would *not* want to use a tree as a data structure?

5. [3 points] Consider the following statement: "a red-black tree will always have alternating colors at each level of the tree (i.e, all nodes of height 0 are black, all nodes of height 1 are red, etc.)". Is this true or false? Briefly, why?

6. [3 points] Briefly explain why fixing the lowest imbalanced node of an AVL tree will necessarily re-balance the entire AVL tree.

7. [3 points] Which is better: an AVL tree or a red-black tree? Briefly, why?

Page 4: Hashes

8. [3 points] Finding an item in a hash table (using quadratic probing) usually takes constant time, but its worst-case run time is $O(n)$. Briefly describe a scenario that would lead to a linear running time for a search in a hash table.

9. [3 points] List the three requirements of a hash function. For each requirement, briefly describe why it is necessary.

10. [3 points] What is the best collision resolution method? Briefly, why?

11. [3 points] Briefly describe two reasons why we make our hash tables of a prime number size.

Page 5: Assembly

12. [6 points] The (incomplete) x86-64 assembly code below should print the integers 0 through 10 in reverse; complete the code. Each blank contains an instruction with either 0, 1, or 2 arguments. You have access to a `print(long x)` function that prints (via `cout`) the passed parameter to the screen.

```
loop:

    ------------
    mov rbx, 10
.L4:

    ------------
    call print

    ------------
.L3:
    cmp  rbx, 0
    jge  .L4

    ------------
    ret
```

13. [6 points] Here is a method in x86_64 assembly; there are three (3!) questions to answer:

```
foo:
    sub     rsp, 8
    mov     [rsp+8], rdi
    add     [rsp+8], rdx
    mov     rax, [rsp+8]
    mov     [rsi], rax
    add     rsp, 8
    ret
```

a. How many parameters are there?

b. For each parameter, are they passed by reference or value? Briefly, how do you know?

c. What does this method do?

Page 6: IBCM and Miscellaneous

14. [3 points] What does it mean that IBCM is Turing complete? If IBCM didn't have the `jmpl` command, would it still be Turing complete?

15. [3 points] Give a compelling example of when we might use big-Omega.

16. [6 points] Consider a program that has three .cpp files: foo.cpp, bar.cpp, and qux.cpp (and .h files for each). Write a Makefile that will successfully compile these three files into an a.out executable. You must use a suffix rule in it! This should not be overly long – you only need about 5 lines for this (more is fine, but don't overdo it).