CS 2150 Exam 1, fall 2018

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: C++

1. [3 points] White a *snippet* of code that will *always* cause a segmentation fault. A snippet is just enough to show the cause – you don't have to worry about `#include` lines, `using namespace std;`, the `main()` declaration, etc.

   Lots of possibilities here. Note that `int *p; cout << *p;` will not *definitely* cause a seg fault, and gets partial credit (but most of the credit). `int *p = NULL; cout << *p;` will *always* cause a seg fault.

2. [3 points] *Briefly* give an example of when you might want the `operator=()` operator override function to be different than the copy constructor.

   deep copy versus shallow copy; assigning a unique ID upon construction but not assignment, etc.

3. [3 points] *Briefly* describe what the `friend` keyword does in C++.

   Allows bypassing of `private` visibility.

4. [3 points] A reference in C++ is just a memory address, as is a pointer. Since the language already had pointers, why were references added?

   Multiple possible answers here. One is to allow for "safe" memory address types that can not crash (think references as parameter types). Another is to allow pass-by-reference.

Page 3: Lists

5. [3 points] Imagine that you had a large *singly* linked list that *might* have a cycle (a cycle is when the ListNodes form a big circle). You can assume that, if a cycle exists, then the *entire* list is that cycle. *Briefly,* how might you *detect* that a cycle exists in your list?

   Keep looping around from the head, and see if you get back there again.

6. [3 points] How might you implement `findKth()`, which finds the k^{th} value, in a linked list in constant time?

   You can't, unless you put everything into an array.

7. [3 points] Give an example scenario in which doubly-liked list has better performance than singly-linked list does. (We are not looking for the extra time needed to assign the second pointer in a doubly-linked list here)

   When one has to move both forward and backward (and both often and roughly equally).

8. [3 points] What do we use Abstract Data Types (ADT) for?

   a type independent description

Page 4: Numbers

9. [4 points] Imagine that you were designing a new floating point type. In addition to the sign bit, you have to assign how many bits represent both the exponent and the mantissa. Why might you put more bits into the exponent? Why might you put more bits into the mantissa?

more mantissa bits means more precision; more exponent bits means more range

10. [4 points] Convert -2150 into a 16-bit two's complement integer, and show your result as a *little*-Endian hexadecimal number. Note that $2150 = 2048 + 64 + 32 + 4 + 2$, and $2048 = 2^{11}$. You *must* show your work to receive credit for this problem!

$2150 = 2048 + 64 + 32 + 4 + 2 = 2^{11} + 2^6 + 2^5 + 2^2 + 2^1$. In binary, +2150 is 0000 1000 0110 0110. To negate, flip the bits (1111 0111 1001 1001) and add 1 (1111 0111 1001 1010). In hex, that's 0xf79a in big-Endian, or 0x9af7 as little-Endian.

11. [4 points] Consider a 24-bit floating point type that has 1 sign bit, 6 exponent bits, and 17 mantissa bits. Other than that, it encodes just as a IEEE 754 number does. What is the maximum (finite) value that this type can hold? You can leave your answer in formulaic form. Also, what is it's (big-Endian) representation for that value? As a hint, if there are e bits in the exponent, then the exponent offset is $\frac{2^e}{2} - 1 = 2^{e-1} - 1$

The sign bit is 0. The mantissa will be all 1's, which sets the mantissa to (almost) 2. The maximum (finite) exponent for a 6 bit exponent field is 62 (since 63 is reserved), with an exponent offset of 31, yielding 31. So the result is $2 * 2^{31} = 2^{32} = 4,294,967,296$ (about 4.3 billion). The encoding is 0 (sign) 1111 10 (exponent) 1111 1111 1111 111 (mantissa), which is 0111 1101 1111 1111 1111 1111 = 0x7dffff.

Page 5: Miscellaneous

12. [3 points] What is/are the difference(s) between an array base name (i.e., `int a[3];` and a pointer (i.e., `int *a = new int[3];`)?

    An array base name cannot have it's address changed.

13. [3 points] Other than syntax, what are the differences between references and pointers?

    cannot be changed; must be initialized upon declaration; implicit dereferencing

14. [3 points] What is the correct command to compile three files (driver.cpp, lib.cpp, and lib.h) and name the output executable `prog`?

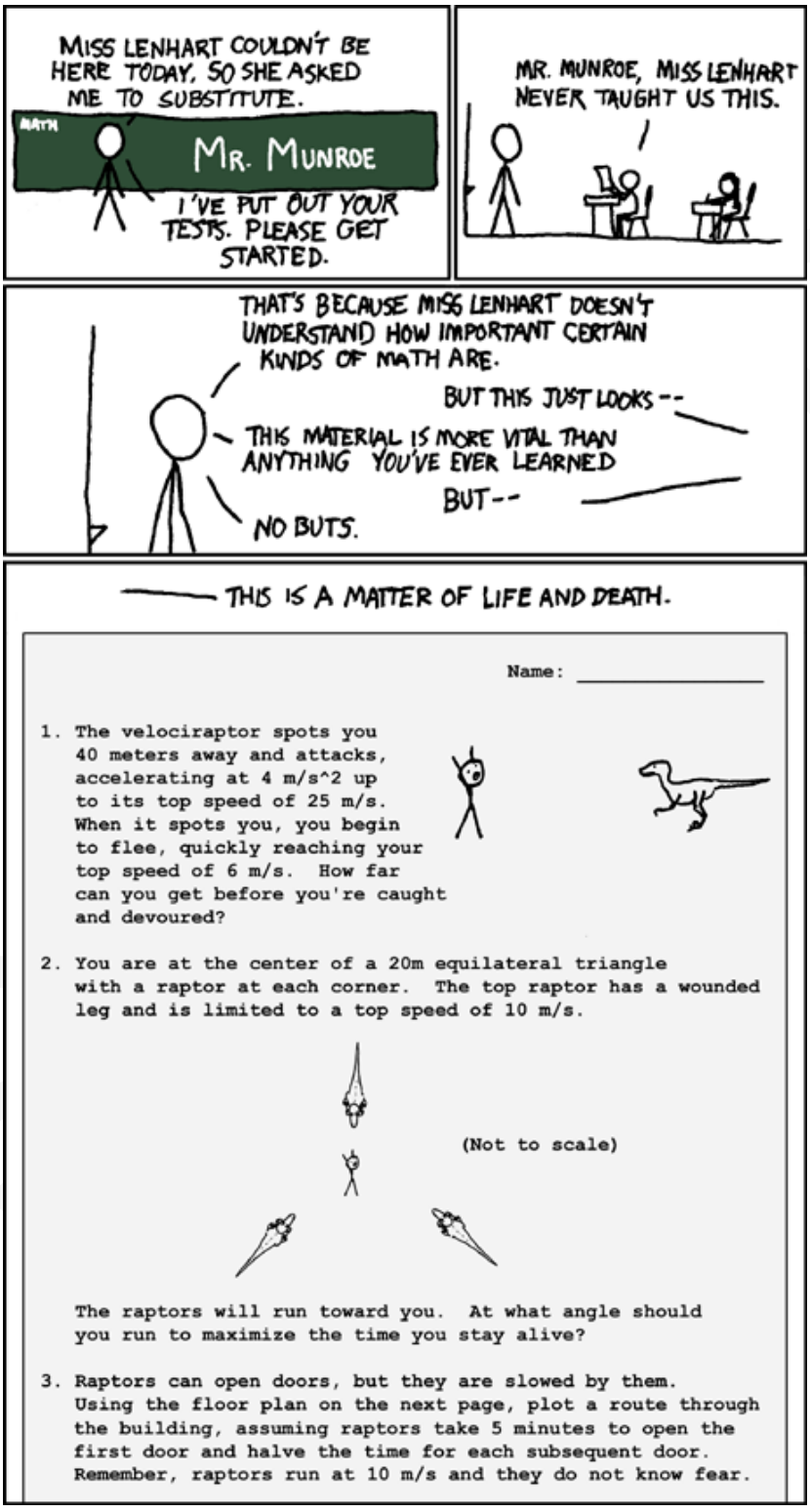
    ```
    clang++ -o prog driver.cpp lib.cpp
    ```

15. [3 points] What is the compiler's pre-processor?

    The first "stage" of the compiler that handles `#define` and #include commands (among others).

Page 6: No questions here

This page unintentionally left blank.



xkcd #135