CS 2150 Exam 1

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.

____

____

____

____

____

There are 10 types of people in this world:
those that understand binary,
and those that do not.

(the bubble footer is automatically inserted into this space)

Page 2: C++, page 1

1. [3 points] Other than syntax, what are the three differences between pointers and references?

2. [3 points] C++ has a special syntax for initialization of fields from a constructor: `ListNode::ListNode () :  next(NULL), prev(NULL), value(0) {` is an example that could be from lab 2. What does this syntax allow that the setting of a field inside the constructor body does not allow?

3. [3 points] The following code segment is legal: `if ( x = 0 ) { ... } else { ... }`. Note that the `if` clause has a single equals sign. What does this statement do, and why?

4. [3 points] When is a destructor called? List two different things that causes a destructor to be called.

Page 3: C++, page 2

5. [3 points] Why would we pass a value by constant reference into a subroutine?

6. [3 points] What is the difference between a shallow copy and a deep copy?

7. [3 points] Why would one write templated code in C++?

8. [3 points] Describe a problem that the `#ifndef/#define/#endif` notation that we discussed in class will solve.

Page 4: Lists and Arrays

9. [3 points] What is an abstract data type? When would we use it?

10. [3 points] What is the most efficient implementation for a stack? Why?

11. [3 points] Diagram what the array defined by `int x[3][4];` looks like in memory. Note that this array has 3 rows and 4 columns.

12. [3 points] Give two differences between how C++ treats an array base name versus a pointer.

Page 5: Numbers, page 1

13. [6 points] Encode -17 as a 32-bit *little-Endian* two's complement number. Your answer should be in hexadecimal format.

14. [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?

15. [3 points] Convert 123 in base 7 to base 3.

Page 6: Numbers, page 2

16. [12 points] Years ago, before the IEEE 754 standard for floating point numbers was published (in 1985), there were problems transferring floating point values, since different machines used different standards. We are going to examine a hypothetical communication of a 32-bit floating point value between a SunOS machine (which used the IEEE 754 standard) and an AIX machine (which used IBM's standard). The IBM standard is the same as the IEEE 754 standard that we learned in class, except that it uses 7 bits for the exponent and 24 for the mantissa. We will assume that both machines are big-Endian. The SunOS machine sends the value 4.5 to the AIX machine, which then decodes the value. What value does the AIX receive? You will need to show your work to receive credit (full or partial).

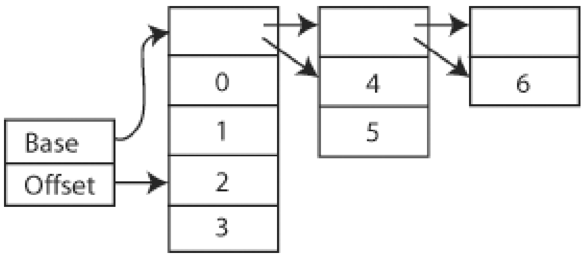
    a. [6 points] Step 1 is to encode 4.5 as the SunOS machine would encode it (i.e. using the normal IEEE 754 format that we discussed in class). Show your work. The result should be left in hexadecimal format.

    b. [6 points] Step 2 is to decode that hexadecimal value from (a) as the AIX machine would decode it (i.e. using 7 bits for the exponent and 24 for the mantissa). Show your work. You may leave the result in rational format, if you prefer.

Page 7: Lirays, page 1

[6 points for reading this] Consider a data structure called an Liray – it's a combination of a list and an array. The liray consists of a series of 'blocks', each of which holds some data in an array, and also a pointer (or two) to the next block. A figure of a liray is shown below.

The size of each successive block decreases geometrically from the previous one – so the first block holds 1/2 the elements, the second one holds 1/4 of the elements, the third one holds 1/8, etc. Note that inserts and deletes may affect this somewhat, but it generally is kept to those ratios (and thus we can assume it is those ratios for our analysis).



Pointing to each block are two pointers: the base and the offset. The base points to the block itself, and the offset points to the first element in that block. Thus, the liray shown here is the list 2, 3, 4, 5, 6. The elements 0 and 1 are not part of the list of elements. Notice that each block has a base and offset pointers to the next block in the liray. Such a block would probably be implemented as an object, with an array to hold the numerical elements and a `base` and `offset` pointer fields to hold the pointers.

Removing the first element in a block (which could be the first element in the list, but could also be element 4 in the liray above) is easy – just make the offset pointer point to the next element in the block. This is how elements 0 and 1 were removed from the liray above, and why the offset pointer in the diagram above currently points to element 2.

Repeatedly removing the first element will eventually remove all the values in a block by moving the offset pointer down an element each time. If we remove 2, 3, and 4 from the liray above, then that block is deallocated, and the base/offset pointers are assigned to point to the next block.

17. [3 points] Give one advantage and one disadvantage of a liray over an *array*. You may want to answer the questions on the next page before answering this question.

18. [3 points] Give one advantage and one disadvantage of a liray over a *linked list*. You may want to answer the questions on the next page before answering this question.

Page 8: Lirays, page 2

19. [3 points] How many pointers will be needed for a liray of size n? Why? Your answer to the first part should be in terms of n.

20. [3 points] What is the worst-case running time of a find in a liray? Why?

21. [3 points] The *average*-case running time of a find is better than the worst case, and not just a decrease in the big-Theta constant. What is the faster running time, and why?

22. [3 points] Briefly describe how an insert of an arbitrary element into the middle of the liray would work. What is it's running time?

Page 9: Complexity analysis

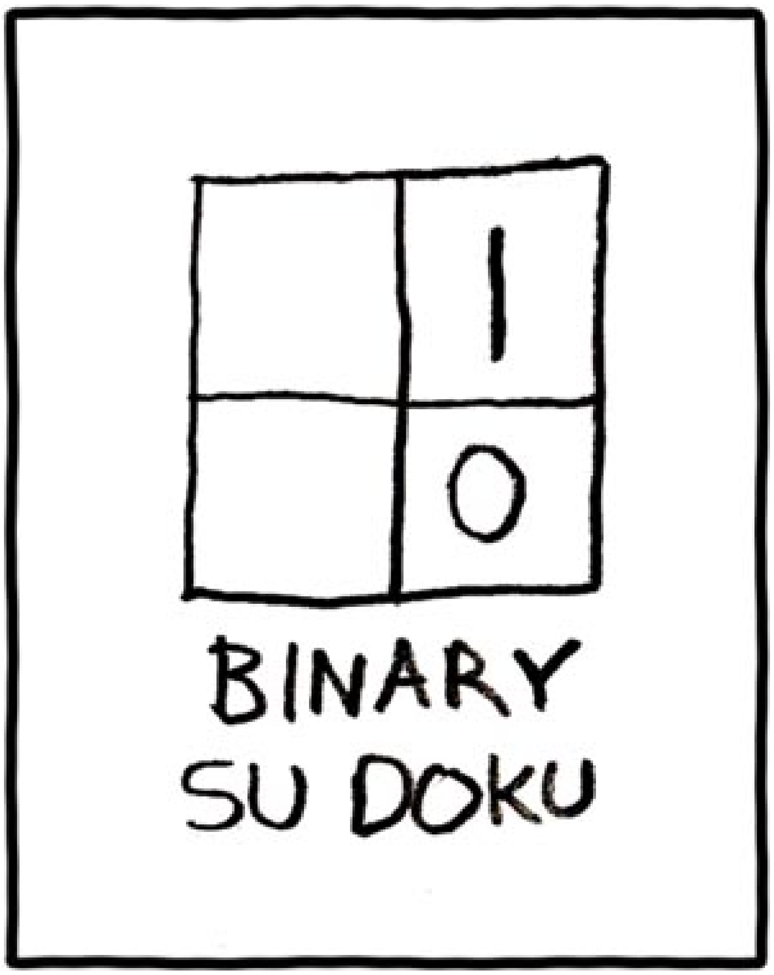
23. [6 points] Prove that $3n^2$ is $\Theta(n^2)$.

24. [3 points] We saw big-Theta in the previous question. Now tell us all about little-theta (how it differs from big-Theta, etc.).

25. [3 points] Why don't we like to use big-Oh and big-Omega, and instead prefer big-Theta?

Page 10: Sudoku!

26. [0 points] Can you complete this sudoku problem? It's quite a challenge! It's not worth any points, though.



(from `http://xkcd.com/74/`)