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. 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!

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.

_____

_____

_____

_____

_____

The Tao that is seen
Is not the true Tao,
until You bring fresh toner.

(the bubble footer is automatically inserted into this space)

Page 2: C++

1. [3 points] Explain precisely (but succinctly!) what is wrong with the following constructor: `Foo() { ListNode* list = new ListNode(); }`. We assume this is from a class `Foo` that has a single `ListNode* list` field.

2. [3 points] Explain precisely (but succinctly!) what is wrong with the following constructor: `Foo() { ListNode temp(); list = &temp; }`. We assume this is from a class `Foo` that has a single `ListNode* list` field.

3. [3 points] Briefly explain why we would want to pass a parameter by constant reference.

4. [3 points] Why does C++ prevent you from changing the address of a array base name? In other words, given `int x[3];`, why can't we change the value of `x`?

Page 3: More C++

5. [3 points] What is the purpose of a destructor, and when are the two times that one is called?

6. [3 points] When is the copy constructor called? When is the `operator=()` method called? Differentiate between when the two are called.

7. [3 points] What does the `friend` keyword do?

8. [3 points] What is the purpose of templates in C++? Why can we not achieve them in the same way Java did before Java created generics?

Page 4: Lists

9. [3 points] Consider the choice between using a vector or a linked list to store elements for some application. Give (and briefly explain) two advantages of using a vector, and two advantages of using a linked list.

10. [3 points] Why do we discuss data structures using Abstract Data Types (ADTs)?

11. [3 points] Give (and briefly explain) two advantages of implementing a stack using arrays (or vectors) and two advantages of implementing a stack using linked lists.

12. [3 points] Why do we care about iterators? What are they good for?

Page 5: Numbers

13. [6 points] Consider the following C++ code. When executed on a 32-bit x86 machine, what is the output?

```
union {
  int x;
  int *p;
} foo;
foo.x = -157;
cout << foo.p << endl;
```

14. [3 points] Convert 43 in base 7 to a base 4 number.

15. [3 points] Why do we need to worry about big-Endianness and little-Endianness?

Page 6: Numbers: floating point

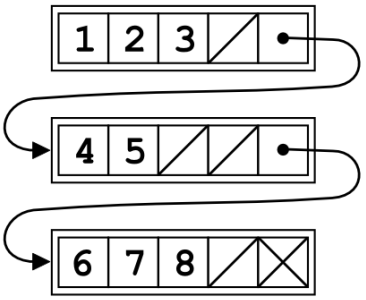
The failure of a Patriot missile to intercept an incoming Scud missile on February 25, 1991 was blamed on a number of factors, once of which was the use of a 24-bit floating point type. We are going to work with this 24-bit float type, which we are going to call a `partial` for this question.

16. [3 points] We assume that the encoding and decoding happens just as with IEEE 754 floating point numbers. Determine reasonable values for the number of bits in the exponent (and thus in the mantissa), and the exponent offset. Give a short argument as to why your chosen number of exponent and mantissa bits is a valid choice.

17. [4 points] What is the minimum and maximum value of the `partial`? You can leave your answer in a power-of-two formula.

18. [5 points] Convert 17.625 to a `partial` floating point number; your final answer should be in big-Endian hexadecimal form. Show your work!

Page 7: Fraglists



[6 points for reading this] Consider a new data structure, called a fragmented list, or *fraglist* for short. A fraglist is a list split into *segments*, where each segment can hold up to m elements. Each segment has a `next` pointer to the next segment in the fraglist. One can consider a fraglist to be an array or vector that is split into parts.

In the fraglist to the right, there are 3 segments, and each segment can hold up to $m = 4$ values. The cells with a slash through them do not have elements, and the right-most cell is the `next` pointer ("X" for the `next` pointer indicates `NULL`).

One stores elements in a fraglist similarly to a standard list. In particular, they can be sorted or not – but like most lists, we assume they are not sorted unless specified otherwise.

Any segment in a fraglist can hold up to m elements, but no fewer than $m/2$ elements (the only exception to this is if the fraglist has only one segment – in that case, it can hold fewer). If a segment falls below $m/2$ elements, then the next element from the neighboring segment is transfered over. If such transfer would lower a neighboring segment to fewer than $m/2$ elements, then the values in two segments (which are both $m/2$ or lower) are combined into one, and the unused segment is removed.

Each segment of a fraglist thus holds three things: the number of elements in this segment, the array of elements, and the next pointer. The number of elements in each segment is not explicitly shown in the diagram above. Note that m does not change during the lifetime of the data structure; it is fixed at the time of the data structure's creation.

The following 6 questions deal with fraglists.

19. [3 points] Given a fraglist after many operations, how much unused space will there be in each segment? In other words, how many elements will the average segment hold?

20. [3 points] What will be the running time of an insert operation into a fraglist? Why? If it is in the same complexity class, will it be any faster? Why?

Page 8: Fraglists, continued

21. [3 points] What will the running time be of a delete operation, assuming that the element has already been found? Why? If it is in the same complexity class, will it be any faster? Why?

22. [3 points] What is the running time of the findKth operation (i.e., finding the 10th element)? Why? If it is in the same complexity class, will it be any faster? Why?

23. [3 points] Give (and briefly explain) an advantage and a disadvantage of fraglists as compared to vectors.

24. [3 points] When would we want to use a fraglist over a vector in an implementation? Your answer for this can not be a repeat of your previous answer.

Page 9: Miscellaneous

25. [3 points] Why do we want you to not use an Integrated Development Environment (IDE) in this course?

26. [3 points] Why do we find big-Oh and big-Omega not all that useful in the context of this course?

27. [3 points] Draw a diagram of what `int x[3][3];` looks like in memory.

28. [3 points] Briefly describe what the three UNIX pipes do, and give one example.

Page 10: Comic!

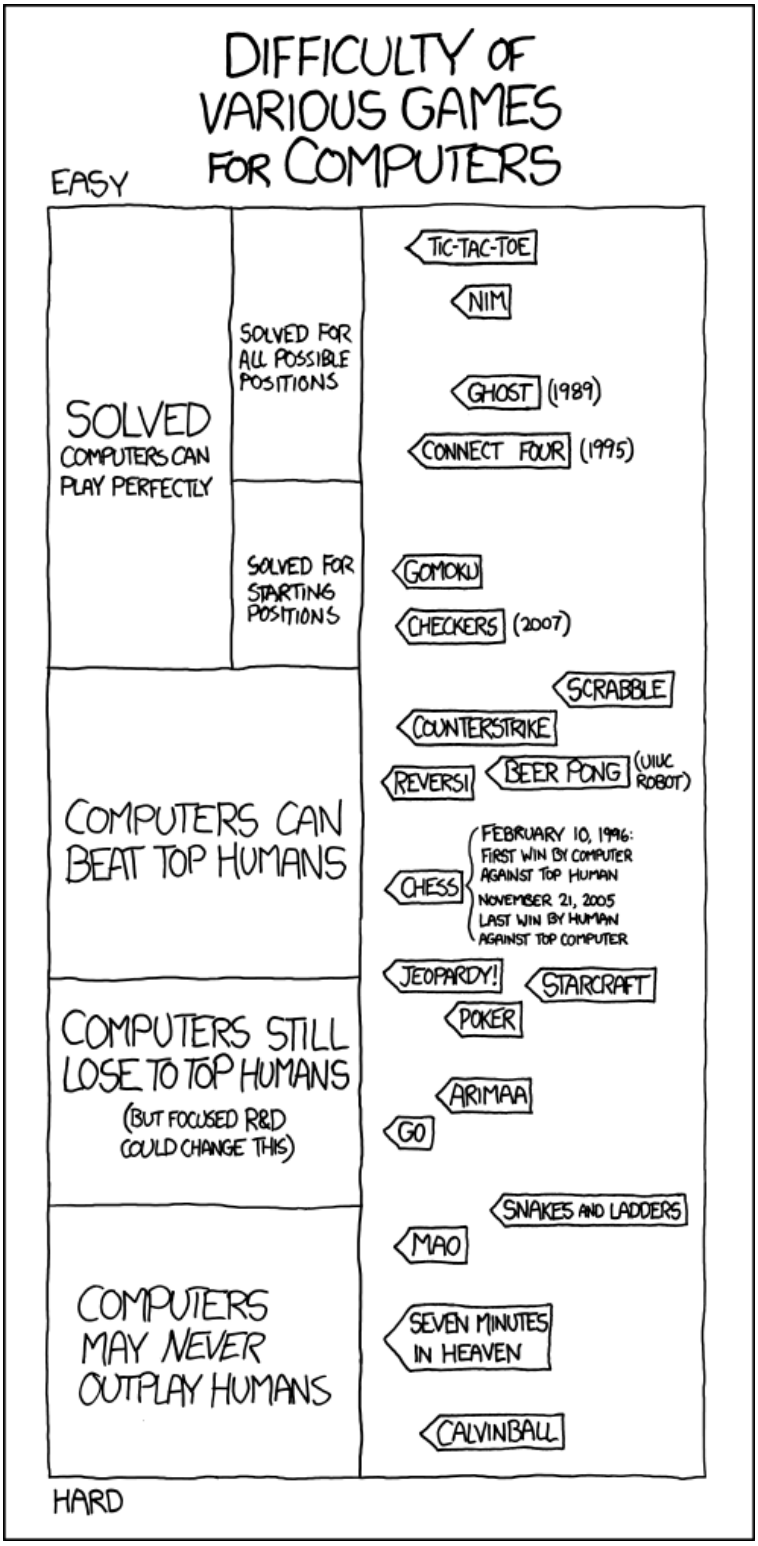


Figure 1: http://xkcd.com/1002/