CS 216 Midterm 1

You MUST write your name and e-mail ID on EACH page and bubble in your userid at the bottom of EACH page – including this page.

If you do not do this, you will receive a zero for that page! (Or a grade penalty if you leave out the bubble form for this page)

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 forms. So please do that first. Sorry to have to be strict on this…

Other than bubbling in your userid at the bottom, please do not write in the footer section of each page.

There are 8 pages to this exam – once the exam starts, please make sure you have all 8 pages.

There are three types of questions: short, medium, and long answer. You can tell the difference because the short answer have a quarter page to answer, the medium have half a page to answer, and the long have a full page to answer the question. The short answer questions should not take more than a line or two to answer, and are worth 4 points each. *Your answer should not exceed about 20 words.* The medium answer questions are worth 8 points each, and the long answer questions are worth 15 or 20 points each. There are 98 points of questions; the remaining 2 points you get for filling out the bubble form on this page, to yield 100 points. There is 105 minutes (1 hour 45 minutes) to take the exam, which means you should spend about 1 minute per question point; the extra 5 minutes are for bubbling in the exam footer.

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 here:

____

____

____

____

There are 10 types of people in the world – those that understand binary and those that don't.

---

1. [4 points] What does the vertical bar character (i.e., ‘|’) do in Unix?

2. [4 points] What does the Unix `chmod` command do?

3. [8 points] Given the C++ code: `int x[3][2] = { {1,2}, {3,4}, {5,6} }`, draw a diagram showing how this array is stored in memory

---

4. [4 points] Not counting the C++ syntax (i.e. the punctuation characters used), list three ways in which references are different than pointers.

5. [4 points] Other than a postfix calculator, what are three applications of the stack data type?

6. [4 points] What does the `explicit` keyword do in C++? What about `friend`?

7. [4 points] On the right side of the footer are five fields: department, course, exam, page, and version. They are encoded in binary in column-major order (least significant bit at the top-left). What are the values encoded on this page's footer?

(the bubble footer is automatically inserted in this space)

8. [4 points] List two uses for the C++ `#define` pre-processor statement.

9. [4 points] Can you pass a pointer by reference? When might you want to do this?

10. [4 points] Convert 1121_3 to base 7

11. [4 points] Convert 0x3f2d to octal (base 8)

(the bubble footer is automatically inserted in this space)

12. [20 points] UVa has designed a new computer processor! It's called the Blintel Quentium 6. While it has many unique and different features, the important one (for this problem, at least) is how it handles numbers – any integer value is written to (and read from) memory in one endian format (we'll say little, but it doesn't matter), but any floating point number is written to (and read from) memory in a the other endian format (we'll say big). An int value is 32 bits and is encoded using two's complement; a float value is 32 bits and is encoded using the IEEE 754 standard (what we saw in class). Keeping this in mind, consider the following C++ code:

```
#include <iostream>
using namespace std;
union foo {
   int x;
   float f;
};
int main() {
   foo bar;
   bar.x = 2113;
   cout << bar.f << endl;
   return 0;
}
```

Do not write here – put your answer to question 12 on the next page

What gets printed? Be sure to show all your work!

If you don't know how to answer this question, you can get partial credit by encoding 2113 as a two's complement integer, and decoding 0x419c0000 (which is not what 2113 encoded is, by the way) as a floating point number.

Note: in order to give you a lot of space to work this one out, you should NOT write your answer on this side of the page – instead, use page 6 for your answer.

Do not write here – put your answer to question 12 on the next page

---

(the bubble footer is automatically inserted in this space)

Answer space for question 12

----

13. [15 points] What does memory looks like after the following C++ code is executed? What is printed? If any of the lines would cause a compile-time error, state so, cross that line of code out, and write the memory diagram and program output as if that line were commented out. Since we are only dealing with ints (and pointers/references to such) here, you can write a 'x' in a spot in memory to signify that it is not initialized.

```
int a = 7;
int *b = &a;
int &c = a;
int d[5] = { 1, 2, 3, 4, 5 };
int *e = new int[5];
int *f = NULL;
int &g = *f;
c = b;
c = *b;

cout << "a:  " << a  << endl;
cout << "b:  " << b  << endl;
cout << "*b: " << *b << endl;
cout << "c:  " << c  << endl;
cout << "d:  " << d  << endl;
cout << "e:  " << e  << endl;
cout << "*e: " << *e << endl;
cout << "f:  " << f  << endl;
cout << "*f: " << *f << endl;
cout << "g:  " << g  << endl;
```

(the bubble footer is automatically inserted in this space)

14. [15 points] In lab 3, you received input for the postfix calculator though normal input (probably `cin`). For this problem, you will need to write a *complete* C++ program that receives input for the calculator through command line arguments, and prints out the result. You can assume:
    - Including a "postfixcalc.h" file defines both the stack and the postfix calculator class, PostfixCalc.
    - After each token is read in, you can just call the PostfixCalc's addToken() method to add that token (whether a number of an operator) to the stack. This method takes in a char* as a parameter.
    - You will want to call the PostfixCalc's evaluate() method, which will return the result.
    - You can assume that the command-line input is well formatted.
