CS 445 Final Examination **Name _____**

Answer all twelve questions. This is a closed book, closed note exam.

1. (10 points) In final code for the x86_64 architecture how does a function allocate space for the local variables, including temporary variables, used by that function? Give some example code that might be used to allocate 72 bytes of local space.

2. (10 points) Why are the lexical analysis rules for class names in 120++ more complicated than the lexical analysis rules for variable names? What special cases did you have to account for when handling class names in 120++?

3. (20 points) Write a regular expression that will match Visa credit card numbers. They always of digits from 0 to 9 and always begin with a 4. They have either 13, 16, or 19 digits total. If they are 16 digits, there are optional spaces between each group of four digits.

4. (20 points) Your compiler semester project required you to integrate several subsystems of varying complexity in order to get working code output. What design and development principles did you apply while working on your project this semester? What did you learn about software design and development from this experience?

5. (20 points) Did you populate your 120++ compiler symbol tables (1) during parsing, or (2) after parsing? Use pseudo-code and/or diagram(s) to describe how you implemented and populated the symbol tables in your compiler.

6. (20 points) How many scopes did you have to handle for the 120++ subset of C++, and what were they associated with? Describe how you organized your symbol tables in order to support the lookup of symbols from differing scopes.

7. (20 points) The subset of C++ that we handled included classes with private member variables and private and public member functions. How did the notion that all member variables are private impact semantic analysis in your compiler? What extra data or computation was necessary in order to support private and public member functions?

8. (20 points) Analyze the C++ code fragment below. Draw syntax trees for the executable statement(s). Report what a compiler's type checker would do in order to determine whether the types were correct. Then report what the outcome of type checking would be.

```
#include <iostream>
using namespace std;
int irand(int);
int k();
int main()
{
   int j;
   j = irand(42);
   cout << j + k;
}
```

9. (20 points) Draw a syntax tree for the following executable statement(s). Generate intermediate three address code (in the form of a linked list diagram) for them.

```
#include <iostream>
using namespace std;
int main()
{
   int i=5, j=7, k;
   cin >> k;
   k = k + i * j / k;
}
```

10. (20 points) Write a pseudo-code outline of an intermediate code generator. What are its input and outputs, and how is it organized? In addition to the main functionality, what auxiliary tasks or helper functions will be needed?

11. (20 points) (A) What are the primary tasks involved in final code generation? (B) How is parameter passing different in final code generation than it was in intermediate code generation? You may focus on the x86_64 architecture in constructing your answer.

12. (30 points) Give an example boolean expression in which the outcome would be different if short-circuit evaluation is used than if the boolean expressions are fully evaluated. Then sketch out the three-address intermediate code for the short-circuit evaluation of your boolean expression.