Project Proposal for “C+ : More than C, Less than C++”

Alexander Stein (as5281) & Eric Johnson (efj2106)

- Effectively Implements all of C
    - Structures, Unions, file-I/O
    - C-like pointer, arrays, strings, references, functions
    - Includes most keywords
        - Auto, double, int, struct, break, else, long, switch, case, enum, typedef, char, extern, return, union, const, float, short, unsigned, continue, for, signed, void, default, goto, sizeof, volatile, do, if, static, while
- Also implements basic objects classes with inheritance and methods
    - Implements **public, private, protected** member keywords
    - Implements **namespace**
    - **No overloading - want to avoid polymorphism as much as possible because this a deep, deep rabbithole.**
- Interesting programs: Dijkstra's Algorithm, Bellman-Ford Algorithm
    - Requires ability to produce data structures
    - Useful to represent complex structures like graphs, perhaps in a standard library
- Project idea: Graph Database
    - Implement lightweight high-efficiency graph database, stored in a data-structure which resembles the database itself, enabling fast search, etc.
- Includes a profiler
    - Since this is in the spirit of C/C++, we want it to be fast. We will include a VERY BASIC profiler to detail the speed which code executes on the (x86) CPU, and what portions of the code are taking the most time.
- Test Driven Development
    - Again, since this is in the spirit of C/C++, the compiler will not do any garbage collecting / checking for out of bounds errors. Our test-bench will look for such things in addition to checking for basic functionality. Will use python unit-tests to automatically run regressions on our language any time a change is made.