Typically, UML diagrams serve as a model from which a final product (a computer program, a database, hardware, etc.) is created. Working from diagrams to product is called forward engineering. For this assignment you will convert a series of small code fragments into UML diagrams, a process called reverse engineering. (A CASE tool that can do both processes is said to be able to do round trip engineering.)

Hints:
- Class-scope variables used to implement relationships between classes are not shown on the class diagrams; the relationships are shown with the appropriate connector symbol between classes.
- The Visio UML Static Structure palette
  - calls inheritance “Generalization”
  - explicitly shows “Binary Association” and “Composition” either of which can be converted to “Aggregation”
    - double-click the connector symbol
    - click the aggregation field
    - select “Shared”

Problem 1

```cpp
class Bar;
class Foo {
    private:
        Bar* bar;
        int time;
    }
};
class Bar {
    private:
        Foo* foo;
        double score;
    }
};
```

Problem 2

```cpp
class Employee {
    private:
        string name;
        int id;
        Address* addr;
    }
};
class Address {
    private:
        string street;
        string city;
        string zip;
    }
};
```
Problem 3

class Employee { };

class SalariedEmployee : public Employee { };

class SalesEmployee : public SalariedEmployee { };

class WagedEmployee : public Employee { };

Problem 4

class Answer;

class MultipleChoice {
private:
  Answer* answers[5];
  int correct;
};

class Answer {
private:
  MultipleChoice* problem;
  string answer;
};

class MultipleChoice;

Problem 5

class OxygenSensor { };

class Carburetor {
public:
  void adjust Timing(OxygenSensor sensor) { } }
}

Problem 6

class StringTokenizer { };

Class Database {
public:
  void parseData(string data) {
    StringTokenizer parser(data);
  }
};
Problem 7

class Electron {};

class Proton {};

class Neutron {};

class Nucleus {
private:
    Protons protons[100];
    Neutrons neutrons[100];
    int numberOfProton;
    int numberOfNeutrons;
};

class Atom {
private:
    Electron* electrons
    Nucleus nucleus;
};

Problem 8

class Foo;
class Bar;

class FooBar {
private:
    Foo* foo;
    Bar* bar;
    int count;
};

class Foo {
private:
    FooBar* links[100];
};

class Bar {
private:
    FooBar* links[100];
};