CSCI-1200 Data Structures — Fall 2014
Homework 10 — Multiple Inheritance & Exceptions

For this assignment you will build a class inheritance structure to match the hierarchy of members of the plant and animal kingdoms. Your finished program will read information about organisms from a file, determine each organism's type from a collection of facts, and then output various categorizations of the organisms to another file. For the final part of the homework we will use a somewhat quirky method to deduce the organism type. We will pass the collection of facts to each specialized organism constructor in turn, and if the constructor *doesn't* fail, then we will know we have chosen the correct organism type. Note: The only way for a constructor to fail is to throw an *exception*. Lectures 23 & 24 will be helpful for this homework.

Organism Hierarchy

You are required to recognize 11 different specific organisms: `Bat`, `Bear`, `Cardinal`, `Penguin`, `Platypus`, `Redwood`, `Rhinoceros`, `Tiger`, `Tortoise`, `Swordfish`, and `VenusFlyTrap`. In addition, you will also define 10 intermediate classes: `Animal`, `Bird`, `Endothermic`, `Ectothermic`, `Fish`, `LaysEggs`, `Mammal`, `Organism`, `Plant`, and `Reptile`. Note that a particular organism may be correctly labeled by more than one of these names. For example, a `Tiger` is also a `Mammal`. *Important Note:* Animal and plant kingdom geeks know that there are exceptions to some of the relationships in the hierarchy and may argue that facts or terms used in the input & output files are overly simplified. For the main homework, please follow these examples exactly.

For the first part of the homework you will draw a detailed diagram of the class hierarchy. The diagram must include all 21 different organism labels. You should draw arrows indicating all of the inheritance relationships. To receive full credit, the diagram should be legible, neat, and well organized, with no messy scribbles or cross outs, have a consistent (up or down) orientation to the edges, and contain few or no arrow crossings. Once you have worked out the relationships and structure, you will most likely need to redraw this figure (at least once), to create the final layout. You do *not* need to include all of the member variables and member functions in this diagram, but you should start thinking about where you will store and initialize common data and the appropriate place to define key functions to reduce redundant storage and copy-pasting function implementation. You should label the virtual inheritance paths (described on the next page).

Since many of you will draw this using pen/pencil & paper, this part of the homework is due in hardcopy to your graduate TA ***in your normal lab section on Wednesday, December 3rd***. Even if you choose to draw the diagram electronically, you are still required to print it out and submit the paper in your normal lab section. You may not use late days for this portion of the assignment.

Provided Code

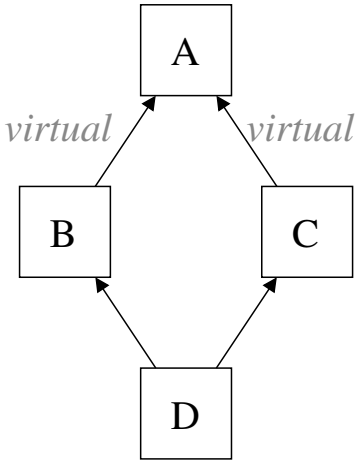
We have provided code that implements all of the I/O for this homework assignment. The executable expects two command line arguments: the input and output files. The input file contains a collection of facts for each organism. Each organism begins with the keyword "`organism`". A fact is a pair of two strings: an organism *property* and the *value* of that property. One property of every organism instance is its *name*. Some of the properties are listed multiple times with different values. For example a `Bear` is known to *eat* bugs, plants, fish, and deer. The fact list for a specific organism instance is not exhaustive, but you may assume the list is detailed enough to uniquely describe a single specific organism type from our tiny subset of the world's creatures. The program's output presents the membership of the input in the 21 organism classes (# of members in each class and the names of the members in that class). The output also lists data on the organism's dietary habits (carnivore, herbivore, or omnivore) and environment (air, land, and/or water). Note: These properties are defined for the organism class and may not be explicitly included as a fact in the input file for that organism instance (because the facts listed for a specific individual will be incomplete).

Important Note: For the base homework, you should not need to modify the provided code. We will be compiling and running your program twice, once with the provided `main.cpp` and once with your `main.cpp` that includes your creative extensions (for extra credit).

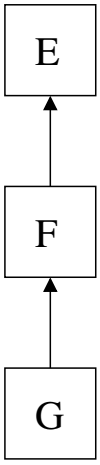
Class Implementation

Your second task is to implement the 21 organism classes. Start with the `input_with_types.txt` file, which explicitly includes the *type* as a fact for each organism. Your solution should contain at least one new `.h` file and at least one new `.cpp` file. You should have a file named `organism.h` that includes all of the class declarations *or* if you use multiple `.h` files, `#include` all of the other `.h` files within `organism.h`. The constructor for each class takes a single argument: the `std::map` of the facts about that instance. We do not need to define the default constructor, the copy constructor, or the destructor. If you get errors indicating the default constructor is not defined, you probably forgot to explicitly pass this data from the derived class to one or more parent classes. In organizing your code for this assignment, avoid unnecessarily duplicating code. For example, don't implement the `livesInWater` function (or variable initialization) in *every* class. Instead, when possible, allow the derived class to rely on the implementation of that function in a parent class. Similarly, don't re-initialize a variable in the derived class if the parent class can initialize it correctly.

The inheritance diagram of these organisms includes *multiple inheritance*. Furthermore, the multiple inheritance is in the form of the *Diamond Problem*. That is, Class D multiply inherits from Class B and Class C, and Class B and Class C each inherit from Class A. Thus when an object of type D is created, in turn instances of B and C are created, and unfortunately both will try to make their own instance of A. If two instances of A were allowed, attempts to refer to member variables or member functions of A would be ambiguous. To solve the problem, we should specify that B *virtually* inherits from A and that C *virtually* inherits from A. Furthermore, when we construct an instance of D, in addition to specifying how to call constructors for B and C, we also explicitly specify the constructor for A. Note how in the single inheritance example on the right with no virtual inheritance, G only explicitly calls a constructor for F.




```
class A {
public:
  A(int x) {}
};
class B : virtual public A {
public:
  B(int x) : A(x) {}
};
class C : virtual public A {
public:
  C(int x) : A(x) {}
};
class D : public B, public C {
public:
  D(int x) : A(x), B(x), C(x) {}
};
```




```
class E {
public:
  E(int x) {}
};
class F : public E {
public:
  F(int x) : E(x) {}
};
class G : public F {
public:
  G(int x) : F(x) {}
};
```

For the final task of the assignment you will work with the `input_without_types.txt` file and your code must deduce the type of each organism. The provided code guides you in this implementation. The collection of facts is presented to each constructor (generally ordered from more specific organisms to less specific intermediate types). If a fact contradicts what is known to be true for that specific organism type, the constructor should throw an exception. For example, the program will try to create a `Tiger` with the data first, and only if that constructor fails (throws an exception) will the program try to create a `Mammal`.

Submission Details & Creative Extensions for Extra Credit

For extra credit you may expand the class hierarchy in an interesting and/or creative manner. Describe your extensions in your `README.txt` file and include sample input and output files that demonstrate your extensions. *Your code should still work with the provided input.*

You must do the implementation of this assignment on your own, as described in the "Academic Integrity for Homework" handout. If you did discuss the assignment or error messages, etc. with anyone, please list their names in your `README.txt` file.