# COSI135: Fall 2014
# Assignment #3

Before you begin, you should review "curried functions" in Learn You A Haskell: http://learnyouahaskell.com/higher-order-functions. You should be familiar with type classes/instances and datatypes from the first assignment. All are necessary, or at least extremely useful, in completing this assignment. Some review of list comprehension will probably help, too: http://learnyouahaskell.com/starting-out.

This assignment consists of only one file: `context.hs`. You should be able to make all your edits in here.

1. You are given an "context" that consists of a list of arguments. You can think of this as things that exist in or are considered to be true about the current state of the world. That is, the context is the "here and now." You are also given a starter verb, in this case, "fly." If you look at the variable `facts` (this represents your knowledge base), you will see it contains one item, `fly john to_boston`. Add facts to the knowledge base using items from the context (or add objects to the context). You should create additional verbs to handle some of these objects (at least two additional verbs in addition to "fly." Some verbs may require a prepositional adjunct, some may not. Think about what items from the context (or items you might create) would be relevant for the verb you are working with.

2. Next, you should write some functions to do some basic question answering over your knowledge base. You should create **three** out of the four following functions: `qWhere`, `qWho`, `qWhat`, `qWhen`. The functions you create should answer the appropriate question. That is, if `fly john to_boston` is in your knowledge base, and you ask `qWho fly to_boston`, the code should return "John" (you can be as basic or as

fancy about the output as you want—the point is to get the right answer). You should think about what type your functions will need to be in order to accept a function as an argument. Note that the "questions" should be grammatically correct with respect to their arguments. That is, `qWho fly to_boston` is a valid question, but `qWho fly boston` is not, because `boston` cannot be flown, but can be flown to. The following pointers may come in useful:

Type `facts` to view your complete knowledge base. Pay attention to how it is organized and think about what methods you can use to isolate particular elements of the knowledge base and what information you would need to provide.

For certain questions, the answer may take different forms. For example, for `qWhere fly john`, an answer of "Boston" and answer of "to Boston" are both acceptable. Like your verbs, your question functions may need to be polymorphic.

Certain questions may accept more than one semantic type for an answer. e.g. "Where" selects for a location, "who" usually selects for a person (though not always). Consider the typing of arguments that your questions will accept for an appropriate answer. You may create new types if you so choose, and you absolutely do not have to account for all the types provided at the top of the file; they are just there to get you thinking about some of the possibilities. You only have to use the ones relevant for the verbs you are working with or creating.

- For some extra credit, you can try to implement an answer for the question "how," in a form like `qHow john to_boston` (read: "How did John get to Boston?"). And answer for this might be "fly," which is a function/predicate, not an argument. How might you account for this in your code? What role would the argument typing play in determining an answer to "how"? (If you choose not to try to implement this function but provide some thoughts on the preceding questions, you can receive lesser extra credit. If you implement the function successfully, no explanation is necessary.)