Perry Spector

**Problem 2:** What is the difference between an interpreted and a compiled language?  What are the strengths and weaknesses of each?  Give an example of each, and list a situation for each where a developer should choose one over the other.

To understand the difference between an interpreted and a compiled language, its important to know that Interpreters as well as Compilers both translate source code into machine code. However, the difference lies in how they both translate. Interpreters translate the source code line by line, while the Compilers translate the entire source code in one sitting. The strengths that an interpreter has are that they don’t generate intermediate code, as well as the fact that debugging is easier. A strength of compilers is that they can run very fast. A weakness of interpreters, especially compared to this, is that their translation is slower. A weakness of compilers, is that they have to generate intermediate code. This requires more memory, generally. Similarly, they can be difficult to debug. Examples are Ruby and Python, which use interpreters, while C and C++ use compilers. An example of something that would be made using a compiler language, is a video-chat. This is because there will be minimal lagging (faster), so as to maintain the conversation. An interpreter would be used generally for a smaller project that might involve a lot more debugging, because it would be more efficient.

**Problem 3:** Explain the concept of an algorithm in terms that a young child (6-8 years old) could understand.

An algorithm is a bunch of steps that help you achieve what it is that you want to achieve. It has steps that help you to solve a problem. An example of an algorithm is one that would help you buy an iPhone. The very first step is driving to the apple store. The second step is looking at different phones you might want to buy. The third step is to decide which iPhone to buy. The fourth step is to buy the iPhone. These steps aren’t the only steps that could be used, as each step can be broken down into even more specific steps as well.

**Problem 8:** Compare and contrast the following collections types in Python:

* list/array
* dictionary/hash table
* tuple
* generator (*extra credit!*)

How are they similar?  How are they different?

An array is a list of a type of value, indexed starting at 0. Tuples are like arrays, but they are not mutable.  Dictionaries are sets of values matched by a key value. In order to retrieve data from a dictionary, one would have to reference it with a key.

**Problem 11:** In 1 to 2 paragraphs, explain the concept of Inheritance from OOP and why it is useful to programmers.  When possible, use analogies or examples that non-coders can understand.

Inheritance is the idea that one class can inherit the characteristics—methods, definitions, etc.—from another. It is useful because say someone wanted to expand on a defined class of a shoe, they can create classes like Nike and Adidas. These Nike and Adidas classes would inherit the main characteristics of the shoe class, but hold their own specific or special characteristics as well.