CSE 262: Quiz #3  
Due October 21st, 2022 at 11:59 PM

The quiz has TWO questions. Please submit your answer by updating this file in the quizzes folder of your Bitbucket account, and then committing and pushing. You should use as much space as you want for each answer. Please be detailed in your answers. Remember: this quiz is worth 9% of your grade, and you will not receive very many points if you do not give detailed answers.

**Question 1:** In class, we discussed that a compiler for a type-safe language *should* know the shape of its data structures. Using this claim as a starting point, study the JSON file format. Then investigate the Google Protocol Buffer format. How do they address the same problem? How do they address different problems? What are the strengths and weaknesses of each?

Compilers for a type-safe language should know the shape of its data structures to maintain data truthfulness until they die. Therefore, when we transmit data to other applications, we need to preserve the data such as their value and types. Similarly, when we receive transmitted data from other applications, we must know what that data represents to interpret and store it correctly. Since there are many ways to store and interpret information, data formats are used to give the data that we send, such as bytes, its meaning so that it can be interpreted correctly.

Two data formats that are commonly used are JSON and Google Protocol Buffers. To start, I would like to explain the JSON and the Google Protocol Buffers formats. JSON is in a text data format that is human-readable, plain-text, uses key-value pairs, and uses a map data structure that is easy to understand. On the other hand, Google Protocol Buffers uses a binary format created by Google to serialize and deserialize data between different services.

These two data formats address some similar common problems. One common problem that they both address is the transfer of data between systems. These two formats can send their data through their serialization based on their data formats that can be interpreted and used by an accepting recipient. Another common problem that they both address is speed. Both of these two data formats are very efficient in encoding. JSON and Google Protocol buffer make use of effective encoding that is much faster and lighter in file size than XML.

Although these two data formats address some similar problems, they also individually address different problems. For example, JSON is primarily used in web development because its text data provides a way for data to be transferred while maintaining a form that is human-readable; this trait is especially beneficial in web development where humans need to be able to read to interpret the information. On the other hand, Google Protocol Buffer is in binary format, where the data is encoded in a bunch of 0s and 1 which is not human-friendly to read, but it is extremely efficient since the data is compressed better. The binary format allows data serialization and deserialization simpler, smaller, faster, and more maintainable than other data formats such as JSON and XML.

The strengths and weaknesses of JSON and the Google Protocol Buffer come from their data format designs and languages and systems that they support. In general, JSON with its text data format would be better for applications that need humans interacting with the data that is being sent and received. JSON’s human-readability benefit also makes debugging easier. On the other hand, the Google Protocol Buffer that has a binary format, which is much harder to read, is designed to be more efficient for processes that do not require a lot of human interaction with the data. Since the Google Protocol Buffer is much more efficient but not readable, it is used in many large internal services that interchange a lot of data where JSON would be much less inefficient for file size and speed. It is also important to note that JSON supports built-in functions and libraries in almost every language and system, but the Google Protocol Buffer supports only a limited number of languages. In general, these two data formats have their own strengths and weaknesses and should be used depending on the goals and needs of the applications.

**Question 2:** Many modern languages have built-in vector and unordered map (hash table) data types, which are capable of holding arbitrary data types. For example, even before the Go language had proper generics, it still supported generic vectors and unordered maps. Other common data types (double-ended queues, lists, ordered maps, queues, stacks, priority queues, etc...) are not built in. There are many possible reasons. Using your knowledge of data structures and what you have learned in programming languages so far, give at least two reasons why these other data types are less likely to have first-class support. Use different data structures to support your first and second reasons.