Serialization API design

For this serialization design, we created two classes to allow the users import from, Serializer and Deserializer. Serializer could transform a type of data to char*, deserializer will convert it back to what it should be.
The types that were allow to be converted currently contains:
Int, float, Boolean, String, float array, string array, and sockaddr_in.

Serializer:

This Serializer contains the char* as buffer and a size_t as the current position to keep track of everything so far.(So, it could allow append to the string buffer, if there are multiple fields in the class). It includes the methods of writes for all of the data types above. Get() will allow the user to see the char* after they serialized.

Deserializer:

Deserializer has 3 field position keeps tracking of the where it has been Deserilized sofar. One is the length of how long it should be, another one is the pos that is currenly at. The last is the buffer being deserialized already. Deserializer includes the methods of reads for all of the data types above.

It is very simple for user to use, they could just import the file by declaring a new serializer or deseializer.

Then with the class they want to serialize, they could add a method for that object class, and used as serialize() or deserilize(), by taking in the serializer or deserializer that we built and for each field in the class use the serializer and deserializer to writed them as char*. And make sure when user implements the serialize and deserialize, the fields that being written and being read are in the same order.

For Assignment 6, we created four classes just for testing:
They are each for primitive type, String, arrays and socket address.