CIS*2750
Assignment 2
Deadline: Thursday, March 6, 11:59pm
Weight: 16.7%

Module 1

Assignment 2 consists of two modules. To simplify the development process, we will not be updating any Assignment 1 functionality - all the changes will be purely additive.

Assignment 2 modules are:
1. A function for writing a Card object to file
2. A function (possibly with helpers) for validating a Card object

These will be useful in Assignment 3, where we will create a Python GUI for creating class lists from VCard files and modifying them using the library we are building in A1/A2.

You are provided with an updated header file, which contains the complete API for A1 and A2. This header will be used in the A2 test harness for grading.

Module 1 functionality

```
VCardErrorCode writeCard(const char* fileName, const Card* obj);
```

This function takes a Card object and saves it to a file in vCard format. You need to make sure that the text output is in correct vCard format, including line endings. This might seem daunting, but your `cardToString` function might already have a lot of this functionality - i.e. traversing the object links. Feel free to borrow some ideas (and code) from it.

However, please avoid the temptation to directly call `cardToString` in `writeCard`, because `cardToString` has a very narrow and specific job - return a newly allocated string representing a Card in a humanly readable format. It is basically our version of Java's `toString` method, and we use it for debugging purposes.

On the other hand, `writeCard` converts a Card object into a valid vCard file - i.e. *serializes* the Card. So to continue with the Java analogy, this would be similar to implementing the `writeObject` method from Java's `Serializable` interface.

NOTE: when writing the file, **DO NOT** fold the lines longer than 75 characters! The card specification states that lines longer than 75 chars the card specification says that they "SHOULD" be folded, not "MUST". Keeping all your output unfolded simplifies the automated testing of your output.

Return value: function must return `WRITE_ERROR` if writing fails for any reason. You do not need to validate the Card object - e.g. verify property names, etc.. Verification will be done using a different function in Module 2, which will be posted next week. However, you still need to validate the arguments - see pre-conditions in the header. You also heed handle potential write errors - e.g. failure to open file for writing.

A simple way to test this function is:
- Read a valid vCard file into a `Card` struct using `createCard`
- Write this `Card` struct into a <u>different</u> file using `writeCard`

- The newly created output file should be identical to the input file - other than line folding (see above). You can adjust with the order of properties in the input file to make sure that everything comes in and out in the same order.