CIS*2750
Assignment 1
Deadline: Monday, February 10, 11:59pm
Weight: 16.6%

Module 2: Error handling details functions

Description

This module provides additional details on expected error handling functionality.

Most error handling will take place in the `createCard` function, which has the following prototype:

```
VCardErrorCode createCard(char* fileName, Card** newCardObject);
```

We do not need to fully validate the format (yet). For example, we do not need to verify that the `N` property has 5 values. However, we do need to verify that the overall grammar of the file, as specified in Section 3.3 of the vCard specification, is correct.

So while an `N` property with 4 or 6 values would not be considered "invalid" in Assignment 1, an `N` property with no values would be considered invalid, since every property must have a value.

If `createCard` successfully creates a Card object, it returns `VCardErrorCode` value `OK`.

However, the parsing can fail for various reasons. In that case, the `newCardObject` argument must be set to `NULL` and the function must return the appropriate error code:
- `INV_FILE` is returned if there is a problem with `fileName` argument - it is null, it is an empty string, file does not exist or cannot be opened, file does not have the correct extension, etc..

If the error is in the file contents, you must use an appropriate error code - see below.

- `INV_CARD` is returned if the file can be opened, but **vCard object itself** is invalid. You would return this code if :
  - The file is missing the `begin` and/or `end` tags.
  - The file is missing any of the required properties, i.e. `VERSION` or `FN`
  - The version is not 4.0

If the error is in one of the properties (optional or required), return a property-specific error code instead (see below).

- `INV_PROP` is returned a **property** in a vCard file is somehow invalid, and cannot be parsed correctly. There are too many possibilities to list, but some of them are:
  - The property has no value
  - The property has no `:` separator
  - Property parameter has no value
  - Invalid line terminator
  - etc.

- `OTHER_ERROR` is returned if some other, non-vCard error happens (e.g. malloc returns null).

Error guidelines:
- Always return the earliest error in the file - i.e. the error with the smallest line number. For example, if the file is missing the `VERSION` and, later in the file, has a property with no value, you must return `INV_CARD`, not

INV_PROP. You could always keep track of the line number that generated the error to avoid ambiguity. Since you are parsing your file in order, this should not be difficult.
- On the other hand, if the problem is that the required FN property is missing its value, the error code must be INV_PROP.
- Make sure the error is specific. If the error is in the header TEL property, the error code must be INV_PROP - not INV_CARD. or INV_FILE.

Error Handling

The error handling functionality in your code must return the correct error code. It should not try to "fix" the error in any way. In particular:
- **Do not** print any error messages to the screen. Simply return the error code, and let the "user" of your library - i.e. whatever function called createCard() - deal with the error
- **Do not** ask the user to re-enter any input, e.g. if file name is invalid. Again, that is not the job of the library. The library simply detects and reports the errors.

You must make sure that you free all temporary resources if an error is encountered. For example, if, when you get to the end of a vCard file, you find that the end tag is missing, you will need to free all memory that you may have allocated to a Card object, close the file, and return INV_CARD.

Make sure you validate all function arguments, particularly in the card parser functions. Passing an invalid argument should not crash your code. For example, printing or freeing a null Card struct must be handled gracefully. An attempt to delete a null card using deleteCard() must result in nothing happening. And attempt to print a null card using cardToString() would result in the function returning an appropriate string, e.g. "null".

Similarly, trying to print a non-existent error using

```
const char* errorToString(VCardErrorCode err);
```

must have a consistent, predictable behaviour - and not crash. If the error code is unknown - i.e. it is not defined in the VCardErrorCode type - return the string "Invalid error code".