CIS*2750
Assignment 2
Deadline: Wednesday, February 27, 9:00am
Weight: 15%

Assignment 2 will consist of three modules. To simplify regression testing of our Assignment 1 code, we will not be updating any Assignment 1 functionality.

Assignment 2 modules:

1. A function for writing a Calendar object to file
2. A function (possibly with helpers) for validating a Calendar object
3. A set of "glue" functions to convert a calendar and its components into JSON strings. These functions will be useful in later assignments: they will help us integrate the C library created in in A1/A2 with the server-side JavaScript code in A3/A4.

You are provided with a temporary header file for A2 Module 1 (`CalendarParser_A2temp.h`). This header will be updated when Modules 2 and 3 come out. Once Module 3 is released, I will post the final official header file for Assignment 2, which will be used in the A2 test harness for grading.

Module 1 functionality

```
ICalErrorCode writeCalendar(char* fileName, const Calendar* obj);
```

This function takes an iCalendar object and saves it to a file in iCalendar format. This might seem daunting, but your `printCalendar` function might already have a lot of this functionality (traversing the object links), so you can borrow a lot of ideas (and code) from it. You need to make sure that the text output is in correct iCalendar format, including line endings.

You **do not** need to validate the contents of the Calendar struct - e.g. verify property names, etc.. This will be done using a different function in Module 2. However, you still need to validate the arguments `fileName` and `obj` - see pre-conditions in the header. You also need to handle potential write errors - e.g. failure to open file for writing.

Return value: function must return `OK` if the file is successfully created and written to disk, and `WRITE_ERROR` if writing fails for any reason.

Please avoid the temptation to directly call `printCalendar` in `writeCalendar`. The reason is that `printCalendar` has a very narrow and specific job - return a newly allocated string representing an iCalendar 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, `writeCalendar` converts a Card object into a valid iCalendar 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.

NOTES:

- This function must always create a new file with the specified name. If a file with this name already exists, it must be overwritten.
- Do not fold the lines longer than 75 characters!. The calendar specification states that lines longer than 75 chars the calendar specification says that they “SHOULD” be, not “MUST”. Keeping all your output unfolded simplifies auto grading of your output.

A simple way to test this function is:

- Read a valid iCalendar file into an Calendar object using `createCalendar`
- Write this object into a different file using `writeCalendar`
- As long as the original file didn’t have any comments, the output file should be identical. You can play with the order of properties in the input file to make sure that everything comes in and out in the same order.
- You can also call `createCalendar` on your newly created file, and verify that a correct Calendar object is created. This is likely how this function will be evaluated by the A2 test harness.