CIS*2750
Assignment 2 grading instructions

Important notes:
- Make sure you compile and run all code on linux.socs.uoguelph.ca!
- A perfect assignment would get 100 as the test score, and will have no deductions

1 Grading rubric

All functions will be graded using an automated test harness

Module 1		*32%*
`writeCalendar()`	32%	
Module 2		*35%*
`validateCalendar()`	35%	
Module 3		*33%*
`dtToJSON()`	4%	
`eventToJSON()`	6%	
`eventListToJSON()`	5%	
`calendarToJSON()`	5%	
`JSONtoCalendar()`	5%	
`JSONtoEvent()`	5%	
`addEvent()`	3%	
Total:		**100%**

Additional deductions:
- You will lose marks for run-time errors and incorrect functionality. Additional deductions include, but are not limited to:
- Any compiler warnings: -15%
- Any memory leaks: -15%
- Any memory errors other than leaks, e.g. under-allocating memory, using uninitialized pointers, etc.: -15%
- Incorrect directory structure: -5%
- Incorrect output filenames created by makefile: -5%
- Any additional failures to follow submission instructions: -5%

Remember to apply late penalty deductions - see the end of the document.

Any compiler errors: automatic grade of zero (0) on the assignment.

2. Test harness instructions

Running the main test harness

The test harness directory structure is:
- `bin` - will contain all executable files
- `src` - contains test cases. Do not modify these in any way.
- `include` - contains test harness headers, as well as `LinkedListAPI.h` and `CalendarParser.h`. Do not modify these in any way.

- `studentCode` - student `.c` files go here.
- `studentInclude` - student `.h` files go here
- `testFiles` - contains various broken and valid vCard files

Running the test harness

- Place your `.c` files into `studentCode`.
- Place your additional headers into `studentInclude`. **Do not** use your version of `CalendarParser.h` and `LinkedListAPI.h`!
  - Make sure you don't accidentally use all `.h` files that the student provided. Doing so may result in the assignment being tested incorrectly.
- To compile and execute the harness, type `make runCalTestA2`. This will clear the bin/ directory, compile the A2 harness, and execute it.

Checking for memory leaks

- Compile and run by typing `make leakTestA2`

This will execute valgrind 8 times. There are 4 tests first create/write/delete, then 4 for create/validate/delete. Each test must show
`" in use at exit: 0 bytes in 0 blocks".`

If there are more than 0 bytes at exit - i.e. if there are leaks - you will see something like:

```
LEAK SUMMARY:
==31556==    definitely lost: 112 bytes in 1 blocks
==31556==    indirectly lost: 20,929 bytes in 681 blocks
==31556==      possibly lost: 0 bytes in 0 blocks
==31556==    still reachable: 0 bytes in 0 blocks
==31556==         suppressed: 0 bytes in 0 blocks
```

Checking for memory errors

- Compile and run by typing `make memErrTestA2`

This will execute valgrind 8 times. There are 4 tests first create/write/delete, then 4 for create/validate/delete. Each test must show

```
==14173== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==14173== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```