# Assignment 2 grading scheme

### 1. Grading scheme

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%** |

There will be no new Makefile requirements for Assignment 2, and your Makefile will not be graded. However, make sure you use the correct directory structure for your submission. See Section 5 for details.

### 2. Upgrading your Assignment 2 grade:

You will have an opportunity to make a minor upgrade to your A2 grade by resubmitting it The resubmission procedure will work the same as A1: there will be a dedicated dropbox, and the deadline for A2 resubmission will be the same as A3 deadline.

The final Assignment 2 grade will be calculated using a weighted average as follows:

A2 final grade = 0.7*initial A2 grade + 0.3*resubmitted A2 grade

NOTE: You **must** submit A2 to be able to upgrade it. So submit A2 even if you think you'll get a zero for it - you'll be able to bump up your grade a bit later. However, if you do not submit it, your grade will be 0 and the upgrade option will not be available to you.

### 3. Evaluation notes:

- If you are aiming for part marks, you do not have to implement every single required function. However, you **must** include a stub for every single function listed in Modules 1 - 3. Without the stubs, your submission will not compile with the test harness, and will automatically get a grade of 0.

- `writeCalendar()` will be evaluated by saving a reference Calendar to a file, reading the file into another Calendar object using your `createCalendar()`, and comparing two Calendar objects. As a result, your `createCalendar()` and `writeCalendar()` must be consistent and compatible with each other - your `createCalendar()` must work correctly with the files that are produced by your `writeCalendar()`. Pay particular attention to how your code treats things like the separator

between the property name and property "description" (i.e. optional parameters+values), and make sure your `writeCalendar()` doesn't produce files that your `createCalendar()` cannot read.

- The functions that accept Calendar objects as arguments will be tested on reference Calendar objects. They must not modify the argument Calendar objects in any way.

- Make sure that you get the format of the output JSON strings right - they will be directly compared to reference strings.

- Remember to use IFS to check your code for compliance with the test harness.

### 4. 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%

- Any compiler errors: automatic grade of **zero** (**0**) on the assignment. Your assignment will be graded on `linux.socs.uoguelph.ca`. The flags used to compile it will be: `-Wall -g -std=c11`. Your code must compile with these flags with no errors or warnings.

- Any infinite loops: automatic grade of **zero** (**0**) on the assignment

- Remember that your source code must compile with the test harness. You should also use IFS to verify that your code is compatible with the test harness. See Lecture 1 and course forums for details.

- Remember to test your code thoroughly, and include both basic and complex cases.

### 5. Submission

the submission must have the following directory structure:

| | |
|---|---|
| `assign2/` | - contains the Makefile |
| `assign2/bin` | - should be empty, but this is where the Makefile will place the shared lib files, as well as all the intermediate `.o` files. |
| `assign2/src` | - contains `CalendarParser.c` , `LinkedListAPI.c`, and your additional source files. |
| `assign2/include` | - contains your additional headers. Do not submit `CalendarParser.h` and `LinkedListAPI.h`. |

As always, you must not modify `calendarParser.h` and `LinkedListAPI.h`. The automated test harness will use the `calendarParser.h` files provided in the Assignment 2 dropbox, and the `LinkedListAPI.h`. used in Assignment 1. If you modify them, your code may not compile.

Submit your files as a Zip archive using CourseLink. File name must be be A2FirstnameLastname.zip.

**Late submissions:** see course outline for late submission policies.