CIS*2750
Assignment 1 test harness instructions

Important note: Make sure you compile and run all code on linux.socs.uoguelph.ca!

Test harness instructions

Running the main test harness

The harness tests calendar creation/deletion, the two print... functions, as well as findIndividua() and getDescendants(). It also tests error handling. It does not test for memory leaks - those scripts are separate (see below). As a result, GEDCOM deletion tests only really test for segfaults and other crashes during deletion.

We don't want to rely on potentially broken makefiles/static libs, so the test harness only uses student .h and .c files.

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, `LinkedListAPI.h` and `GEDCOMparser.h`. Do not modify these in any way.
- `studentCode` - your `GEDCOMparser.c`, `LinkedListAPI.c`, and `GEDCOMutilities.c` go here. Do not include any files that are not part of the parser API.
- `studentInclude` - your additional `.h` files go here - i.e. headers for your additional utility functions, hash table (if you used it), etc. Do not put `LinkedListAPI.h` and `GEDCOMparser.h` here!
- `testFiles` - contains various broken and valid GEDCOM files

Copy their `.c` and `.h` files into the appropriate directories as described above, and run the harness.

To run:
- compile by typing `make`
- run `bin/GEDCOMtests`

The output of the test harness contains all the information about the passed/failed tests, as well as the total score - out of 90.

If your code does not compile, fix the errors. Do not modify the provided headers or the test harness.

Checking for memory leaks

- Compile ty typing `make memTest`
- run `make valgrind`

This will execute valgrind 4 times (with 4 different .ged files). 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
```

If your code is buggy, the valgrind output might be messy. If you're only concerned about leaks, ignore all the errors and just look for memory leak summaries.

If one of the valgrind cases crashes, ignore the leaks for that case - crashed code will leak by its nature. However, make sure you look at the output of the other cases. If all 4 test cases crash, you definitely need to fix your `createGEDCOM` and `deleteGEDCOM` functions.