CMPUT 350 Lab 8 Exercise Problems

Rules:

- You can use all course material and man pages, but no other information such as web pages, books, or written notes. Using other information sources during the exercise constitues cheating.
- Your programs must compile without warning using

  ```
  g++ -Wall -Wextra -O -g -std=c++17 ...
  ```

  In case there are compiler warnings or errors, marks will be deducted.
- Test your programs with different values. For now, the speed of your program is irrelevant. So don't spend time on optimization
- You must check for the appropriate preconditions/postconditions. Your program shouldn't crash or have undefined behaviour (**hint**: use asserts)!
- Your programs must be well structured and documented. Use ctrl-x t in Emacs to pretty-print it. Marks are assigned to functionality, program appearance, and comments.
- In case your program hangs, use ctrl-c to terminate it.
- Remember that you need to include the appropriate header files. To find out which ones you need for specific functions such as printf, use then man command.

Submit your solution files `Triang2.cpp` on eClass under "Lab 8 / Submission".

Important: Submit often (the system will only accept solutions submitted before 16:50)

---

1. [29 marks]

In this exercise you will complete a program that generates a Delaunay triangulation by first incrementally generating a triangulation for n points in a bounding rectangle and then flipping edges to increase the minimum interior angle locally.

To get started, download all files into a new folder - individually by right-click-save or by extracing all files from `lab8_ex.zip`

We provide a driver (`disp.cpp`) that animates the process using OpenGL and a working executable (`disp.lab`) as reference. Please note that after downloading this file you have to set its executable flag like so: `chmod +x disp.lab`

Our driver accepts command line parameters that may be useful for testing. Run `./disp.lab -h` to see a list. To reproduce program runs set seed to != 0 (see test scripts `test_diag`, `test_fast`, `test_slow`).

Your task is to complete functions in `Triang2.cpp` in which we also provide hints. Don't change any other file because you'll only submit `Triang2.cpp`.

Start by reading `Triang.h` and `Triang.cpp`, which contain most of the implementations already. We encourage you to use our implemented functions in your solutions.

To compile your program use `make`

You may change `disp.cpp` for testing your functions

You can run `./disp` from a remote machine provided

1. you use x2go, or
2. you are using a UNIX environment on your computer (e.g., MacOS, or hosting Ubuntu on Windows + installing an X-Window server)
3. you have an X-Window server running on your own computer and use ssh -X ... to connect to a lab machine (-X forwards X-Window connections, try to run "xeyes" on the lab machine to see if it works)

IMPORTANT: Please note that any crashes you might experience while testing your functions are most likely caused by your code, which is assumed to be correct by the Delaunay triangulation code that calls them. In such cases, we suggest to test your functions independently.