Title:	Methods and Tools for Software Engineering
Course ID:	ECE 650
Term:	Winter 2021
WWW:	`https://ece.uwaterloo.ca/~rbabaeec/ece650/w21`
Campuswire	`https://campuswire.com/p/G9003B726` Enter 3253 when prompted
Lectures:	Wednesday, 14:30 – 17:20 EST
Instructor:	Reza Babaee, `rbabaeec@uwaterloo.ca`
TAs:	An Qi Zhang, `an.zhang@uwaterloo.ca`
	Arshdeep Kaur Bal, `arshdeep.kaur.bal@uwaterloo.ca`

Assignment 4 - Due Sunday, March 28, 23:59 EST

The repository for this assignment is available at `gitlab`. Run the following to clone your repository.

`git clone ist-git@git.uwaterloo.ca:ece650-1211/a4/USERID.git`
where `USERID` is the ID you use to login to quest (without @uwaterloo).

For this assignment, you are to augment your code from Assignment 2 to solve the minimal Vertex Cover problem for the input graph. Your approach is based on a polynomial time reduction to CNF-SAT, and use of a SAT solver. The following are the steps you should take for this assignment.

SAT Solver

We will be using MiniSat SAT solver available at `https://github.com/agurfinkel/minisat`

MiniSat provides a `CMake` build system. You can compile it using the usual sequence:

```
cd PROJECT && mkdir build && cd build && cmake ../ && make
```

The build process creates an executable `minisat` and a library `libminisat.a`. You will need link against the library in your assignment.

Play around with it. Sample files are available in `https://github.com/eceuwaterloo/ece650-minisat`.

Incorporate SAT

Create a polynomial reduction of the decision version of VERTEX COVER to CNF-SAT. We have discussed the reduction in class. It is also available under the name `a4_encoding.pdf` on LEARN. You are allowed to use your own reduction provided it is sound and polynomial-time. Implement the reduction and use `minisat` as a library to solve the minimum VERTEX COVER problem for the graphs that are input to your program (as in Assignment 2).

As soon as you get an input graph via the '`V`' and '`E`' specification you should compute a minimum-sized Vertex Cover, and immediately output it. The output should just be a sequence of vertices in increasing order separated by one space each. You can use `qsort(3)` or `std::sort` for sorting.

Assuming that your executable is called `a4ece650`, the following is a sample run of your program:

```
$ ./a4ece650
V 5
E {<0,4>,<4,1>,<0,3>,<3,4>,<3,2>,<1,3>}
3 4
```

The lines starting with `V` and `E` are the inputs to your program, and the last line is the output. Note that the minimum-sized vertex cover is not necessarily unique. You need to output just one of them.

Marking

We will try different graph inputs and check what vertex cover you output. We will only test your program with syntactically and semantically correct inputs.

- Marking script for compile/make etc. fails: automatic 0
- Your program runs, awaits input and does not crash on input: + 20
- Passes Test Case 1: + 25
- Passes Test Case 2: + 25
- Passes Test Case 3: + 25
- Programming style: + 5

CMake

As discussed below under "Submission Instructions", you should use a `CMakeLists.txt` file to build your project. We will build your project using the following sequence:

```
cd PROJECT && mkdir build && cd build && cmake ../ && make
```

where `PROJECT` is the top level directory of your submission. If your code is not compiled from scratch (i.e., from the C++ sources), you get an automatic 0.

Submission Instructions

You should place all your files at the top of a gitlab repository. The repository should contain:

- All your C++ source-code files.
- A `CMakeLists.txt`, that builds your C++ executable `a4ece650`.
- A file `user.yml` that includes your name, WatIAM, and student number, and the hours you spent on the assignment. Note that *WatIAM* is the user name for your Quest account, e.g. `agurfink`, and a *student number* is an 8-digit number, e.g. `20397238`.

See `README.md` for any additional information.

The submitted files should be at the top of the `master` branch of your repository.