Title:	Methods and Tools for Software Engineering
Course ID:	ECE 650
WWW:	`https://ece.uwaterloo.ca/~wdietl/teaching/2015f/ece650/`
Lectures:	Thursday, 17:30 – 20:20
Instructor:	Dr. Werner Dietl, `wdietl@uwaterloo.ca`, DC 2522
TA:	Parsa Pourali, `ppourali@uwaterloo.ca`

Office hours by appointment. Begin all email subjects with `[ECE650]`.

Assignment 4 - Due Tuesday, November 17, 12:00 noon

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

Download the latest 32-bit version of `zchaff`: `http://www.princeton.edu/~chaff/zchaff.html`.

Make the executable and the `libsat.a` file. Play around with it. A sample input file is on Learn. Write a simple C program to understand how to use the `libsat.a` library.

Incorporate SAT

Polynomially reduce the decision version of Vertex Cover to CNF-SAT. I provide such a reduction on Learn. You are allowed to use your own reduction provided it is sound and polynomial-time. Implement the reduction and use `zchaff` 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)` for sorting.

For example, on input "`V 5`" and "`E {<0,4>,<4,1>,<0,3>,<3,4>,<3,2>,<1,3>}`", your output should be "`3 4`" in a single line. 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 not try any incorrectly formatted input.

- Marking script for uncompressing/compile/make etc. fails: automatic 0
- Your program runs, awaits input and does not crash on input: + 20
- Passes Test Case 1: + 40
- Passes Test Case 2: + 40

Submission Instructions

You should place all your files in a single folder (directory), `a4-ece650`. The folder should contain:

- A Makefile. It should contain a target "`all`" that makes the `a4-ece650` executable, and a target "`clean`" to clean all your intermediate (e.g., `.o`) files. If we do a "`make clean`" followed by a "`make all`", then everything should compile from scratch.
- All your C sources, including your code from Assignment 2 that reads and maintains the graph.

Your submission should not contain anything from `zchaff`. We will supply the `zchaff` library against which you will link. You may assume that the library is called `libsat.a`.

You should `tar` and `gzip` the folder using the command: `tar czf a4-ece650.tar.gz a4-ece650`. (To `untar` and `gunzip`, we will issue `tar xzf a4-ece650.tar.gz`.)

You should upload `a4-ece650.tar.gz` to the dropbox on Learn for Assignment 4.