CompSci 260P Fall 2022
Project #0: Getting to Know the Virtual Machine

Due date and time: October 3, 2022 at 3:00 PM.
Remember to read the syllabus to be sure you make yourself eligible to be graded for this assignment.

If you were an undergraduate here and took a class from Professor Thornton, or any class from me that had a programming component, the VM should look very familiar to you, as will these setup instructions. However, this is not the same thing as Project #0 from when you were in that class, so you still have some work to do.

In fact, many of this assignment instructions are drawn from his similar assignment for ICS 46 and he was instrumental in helping me set up the VM. That having been said, you should still read these directions, as they may have changed from when you took ICS 45C, even if he was your teacher for that class.

Remember:
● You need to download the VM for this class, not the one for Prof. Thornton’s ICS 46
  ○ If you have an M1, you may use the ICS 46 M1 VM if you’d like.  
    Note that the username/password default on that is ics46/ics46, not compsci260p.
● You need to do this project, not the one for any other class.

For students who wish to work in a quiet environment on campus, I recommend loading the VM onto a USB “thumb” drive and taking it to the ICS third floor computer lab. I also recommend you be very careful in backing up your work in progress, no matter where you are doing this, but especially if you are working via thumb disk in a computer lab.

The Program’s Requirements

As a warm-up, this project asks you to write and submit a short C++ program. The program itself isn’t actually the interesting part, though it’s one that you might find takes you a little bit of time to write. The main goal here is to be sure you’re able to use the VM to do your work, that you learn what you need to know about one of the available text editors to write your program, and that you use the provided tools to gather your files for submission. Even if you normally prefer a different working environment, you would be well-served to use the VM for this project, to be sure that you can use it for your work later in the quarter. Also, if your project does not compile and run in the VM, we will not be able to grade it, and that will cause you to get a zero. Every quarter this happens to some students who, to put it gently, are not pleased with their grade on any such project. Please do not let this happen to you!

See the lab manual for information on how to set up the VM and how to start a project for this class.

For more information about grading, see the relevant section of the lab manual.

1 You know how I said that if you get significant but allowable help from someone else on an assignment, you should cite who and what clearly? That rule applies to me too.
The Program’s Requirements

The Traveling Salesperson problem is as follows: we are given a graph with \( n \) vertices. Our goal is to find the smallest-cost Hamiltonian Cycle in the graph. A Hamiltonian Cycle is a cycle that includes every vertex exactly once and returns to the starting point.

For this project, you are given the graph in the form of an adjacency matrix: a 2D vector where element \([i,j]\) is the cost to travel from vertex \( i \) to vertex \( j \). You are guaranteed that every Hamiltonian Cycle has positive cost, and none have cost more than UINT_MAX.

To accomplish this, you need to implement two functions in `proj0.cpp`:

- **std::vector<unsigned> tsp_bruteforce(const std::vector<std::vector<long>> & graph, unsigned n)**
  - Given the graph with the designated number of vertices, return a vector that represents the smallest-cost Hamiltonian Cycle in the graph.
  - In the return value, do not include the starting point twice. That is, the size of the vector you return should be exactly \( n \), and should be a permutation of the integers \([0,n-1]\).

- **long costOfJourney(const std::vector<std::vector<long>> & graph, const std::vector<unsigned> & journey)**
  - Given the graph and a journey, compute the cost of this permutation for a TSP solution.
  - The last element of the journey vector is not the starting point. Your return value should reflect returning to the starting point as well.
  - You may wish to call this function from `tsp_bruteforce`.

You may assume all inputs are valid; for example, in `costOfJourney`, the second parameter will always be a permutation of the unsigned ints in the range of \([0, n-1]\), where \( n \) is the number of vertices in the given graph. You do not need to do bounds checking or the like.

When we grade this, any test case that takes more than two minutes to run on the instructor’s (reasonably modern) computer may be deemed an unsuccessful test case. We will only choose test cases that can reasonably run in this time; for example, there will be no \( n=50 \) cases: no matter how good a programmer you are, the \( n=50 \) case isn’t going to run in our time constraints. We’ll deal with larger cases in later assignments with different techniques.

I encourage you to read the starting point code given in `proj0.cpp` and the test cases described in `gtest/start.cpp` and ask any questions you may have on EdStem or in office hours.

**Deliverables**

See the Lab Manual for information about submitting projects. We will grade only what was submitted before the deadline. If you replaced some of your files with newer versions before the deadline, we will grade only the most recent submission of each unless you fill out the relevant form before the submission deadline.

We will not grade files submitted after the deadline has passed, nor will we grade files submitted via email or in paper form.
You are responsible for submitting the version of your project that you want graded. We will grade only what you submitted before the deadline. Accidentally submitting the wrong version, or forgetting to submit files, will not be considered grounds for a regrade.

This project is not included in the late work policy for this course. It needs to be completed on schedule and submitted before the due date above. Submissions beyond that deadline will not be considered.

**Your grade on this project**

There are 2 points possible on this project; they are only available by test cases. We will run test cases with the code you submit; each test case is worth some fraction of the grade. Test cases that take longer than two minutes to run on the instructor’s (reasonably modern) computer may be deemed incorrect runs, even if a longer amount of time available to them would cause a correct answer.

For more information about grading, see the relevant section of the lab manual.