For this assignment, you are to:

• Modify the output format of your Python script from Assignment 1 to match the input format

of your C++ program from Assignment 2.

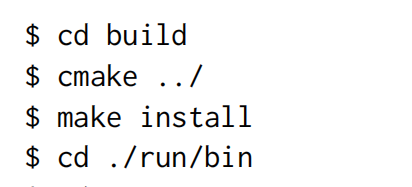
• Modify your C++ program from Assignment 2 to output the input graph on standard output.

• Write a program in C++ to generate random input for your Python script.

• Write a driver program in C++ that uses Inter-Process Communication (IPC) to link the

output of the random input generator to the input of the Python script, and the output of the

Python script to the input of the C++ program from Assignment 2.



$ cd build

$ cmake ../

$ make install

$ cd ./run/bin

$ ./ece650-a3 -s 5 -n 4 -l 5

In the above, the first three lines make your executable, and run the driver program with some command-line arguments.

terminate gracefully (and quietly) once you see EOF at stdin.

Of course, we may omit command-line arguments (see below for what to do in such cases), and specify vertex IDs with s that do not exist, or between whom a path does not exist. The latter two cases should cause your program to report an error.

**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 a3 && mkdir build && cd build && cmake ../ && make install

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 in the a3 directory in your GitLab repository. The directory should contain:

• A CMakeLists.txt, that builds all of the C++ executables: rgen, ece650-a2, and ece650-a3.

**Random input generator**

大小可能需要用if限定一下？

• -s k — where k *≥* 2. The number of streets should be a random integer in [2*,* k].

default of sk = 10

• -n k — where k *≥* 1. The number of line-segments in each street should be a random integer in [1*,* k]. Default of nk = 5.

• -l k — where k *≥* 5. Your process should wait a random number *w* seconds,

where *w* is in [5*,* k] before generating the next (random) input. Default: k = 5.

• -c k — where k is an integer *≥* 1. Your process should generate (*x, y*) coordinates such that every *x* and *y* value is in the range [[k*,* k]. Default: k = 20.

Also, note that your random generator could go into an infinite loop looking for a valid specification. You should disallow this from happening by limiting the number of tries. That is, if your random generator fails to generate a valid specifification for a continuous *A* number of attempts, it should exit() with an error message reported on stderr. A reasonable *A* to adopt may be 25.

Whatever *A* you adopt, your error message should mention it. That is, your error message should be something like, “Error: failed to generate valid input for 25 simultaneous attempts”.

Before adding a new street graph specifification to your Python script, your generator should issue “rm” commands to your Python script to remove all previous streets and replace them with the new street specifification.

After generating the input, the generator must issue the “gg” command.

Thus, a typical interaction of the random generator is as follows:

1. issue enough rm commands to clear any existing street database;

2. issue add commands to add new streets satisfying the specifification;

3. issue a gg command;

4. wait for specifified number of seconds and repeat.

**Driver**

It should send normal output to stdout, error output to stderr, and take input from stdin.