**Name of the group:** sim-city-1

**C1.1: The implementation corresponds to the selected topic and scope. The extent of project is large enough to accommodate work for everyone (2 p)**

The project corresponds to the given topic and implements the features requested in the task description: 2p

**C1.2: The class structure, information hiding and modularization is appropriate, and it is explained and justified in documentation. The file structure corresponds to the class structure (2 p)**

The class structure is mostly good. Class names "vertex" and "edge" are not very descriptive in my mind, instead of e.g. "Road" and "Building", which could be an abstract subclasses for different kinds of road blocks or buildings. So there could have been more room for leveraging inheritance and other object oriented constructs: 1.5p

**C1.3: Use of at least one external library (in addition to C++ standard library). Comment the appropriateness of libraries and their use. (2 p)**

SFML is used appropriately: 2p

**C2.1: Git is used appropriately (e.g., commits are logical and frequent enough, commit logs are descriptive) (2 p)**

Git is used well, and the commit messages describe clearly what had been done: 2p

**C2.2: Make or Cmake (recommended) is used appropriately. The software should build easily using these tools without additional tricks. Nevertheless, instructions for building the project should be provided (1 p)**

Make is used and it works: 1p

**C2.3: Work is distributed and organised well, everyone has a relevant role that matches his/her skills and contributes project (the distribution of roles needs to be described) (1 p)**

The group seems to have worked well together, and work is rather well distributed: 1p

**C2.4: Issue tracker is used appropriately to assign new features and bug fixes (1 p)**

Issue tracker is used well: 1p

**C2.5: Testing and quality assurance is appropriately done and documented. There should be a systematic method to ensure functionality (unit tests, valgrind for memory safety, separate test software and/or something else.) (1 p)**

Testing is well documented, also valgrind is used: 1p

**C3.1: C++ containers are used appropriately (including appropriate use of iterators), and justified (e.g., why certain type of container over another) (2 p)**

Containers are used appropriately where needed: 2p

**C3.2: Smart pointers are used in memory management, describe how (1 p)**

Smart pointers are used well: 1p

**C3.3: C++ exception handling is used appropriately, describe how (1 p)**

The main function contains the "throw" and "catch" keywords, but in a rather curious way. No actual exception classes are used to catch exceptional cases in different classes or functions: 0p

**C3.4: Rule of three / rule of five is followed, describe how (1 p)**

RO3/5 is considered appropriately: 1p

**C3.5: Dynamic binding and virtual classes/functions are used, describe how (1 p)**

Vehicle is a base class for different kinds of vehicles, and dynamic binding is used appropriately with containers. I think Vehicle could have been made pure virtual abstract base class, so that instances of that generic type could not have been created: 1p

**General comments:**

Good project, and the group seemed to have worked well as a team.

**Total points:** 16,5 / 18