ESS201: C++ Programming

Jaya Sreevalsan Nair *
International Institute of Information Technology, Bangalore

Term I: 2017-18 (Lab on 2017-10-24)

(*Note: Lab on 2017-10-23 was an extension of the Lab02b, i.e., lab on 2017-10-17.*)

In this Lab, we will apply operator overloading in the `vector3d` class we have built so far.
Tasks:

1. Work with a copy of `graph.cpp` written for Lab-02b, renamed as `graph_oper_overload.cpp`
2. Overload assignment operator `=` for `vector3d` class.
3. Write new `friend` functions to overload istream and ostream operators (`>>` and `<<` respectively) for class `vector3d` in `graph_oper_overload.cpp`. The streams can use the following format:
   `vector <xvalue> <yvalue> <zvalue>`
   (Note: the angular brackets `< >` need not be there in the output from ostream or while inputting values.)
4. Write new operator overloaded functions for `+=` and `-=` for `vector3d` class.
5. Write new `friend` functions for operator overloading for `+` and `-` for `vector3d` class.
6. Write new `friend` functions to overload istream and ostream operators (`>>` and `<<` respectively) for class `Graph` in `graph_oper_overload.cpp`. The streams can use the format given for sample file at the end of this document.
7. Overload `+` for `Graph`, where the "sum" contains a "union" of the sets of `Point` (stored as private data member in `std::vector` here) and a "union" of the sets of `Line` (stored as private data member in `std::vector` here).
   (Note: The afore-mentioned are set unions. Hence, ensure to remove duplicate occurrences in `std::vector` private data members of the "sum".)
8. Overload `-` for `Graph`, where the "difference" contains a "difference" of the sets of `Point` (stored as private data member in `std::vector` here) of the addends of `Graph` type; and a "difference" of the sets of `Line` (stored as private data member in `std::vector` here)/ `Graph`. In addition to set "differences", ensure that all the edges in `std::vector<Line*>` (private data member of `Graph`) have its nodes in `std::vector<Point*>` (private data member of `Graph`).
   (Note: There can be nodes without any edges attached to them in a "valid" `Graph`, whereas presence of edges without nodes contained in the graph makes it an "invalid" graph. Hence, write a member function `is_valid` for the class `Graph`.
9. Remove functions from the classes which have become redundant owing to operator overloading, and replace the function calls to the redundant functions using the new operators.

*(jnair@iiitb.ac.in)

- Contents in a sample input .txt file, named `graph_in1.txt`:

```
# 10
3.4 2.1 4.2
5.6 9.3 2.2
0.4 8.2 2.3
6.2 0.2 4.2
3.4 0.2 1.3
4.1 4.2 1.4
8.3 9.8 5.2
2.4 0.2 6.8
0.5 9.2 0.1
0.5 0.2 0.1
# 15
0 3
0 5
3 6
4 2
2 0
7 9
4 8
4 9
3 2
7 2
3 5
6 2
5 7
4 7
5 9
# 2
7.4 9.5 4.5
4.3 0.4 1.6
###
```

- Input: Two files in the afore-mentioned format: `graph_in1.txt` and `graph_in2.txt`. Each file contains data for a single graph. Your program should run with file-names input in the command prompt.
  `./a.out graph_in1.txt graph_in2.txt`

- Output: Output sum of the input graphs, followed by newline, difference of the input graphs, followed by newline, followed by output of responses to queries of first file (as instructed in Lab02b), followed by a newline, and followed by output of responses queries of second file.
  (Note: The queries in an input file pertains to the graph data given in the same file. They are not applicable across input files or graphs.)