Lab 11 grading guidelines

Pre-lab

Out of 10 points. This lab part **is** expected to compile.

For the doxygen commenting, they should have something beyond the tags that were in the lab PDF (those tags were @brief, @return, @param, @todo). This will require briefly looking at their code. Note that the doxygen comments will appear as a 'director html' or a 'directory doc' link in the web page that the grading script produces.

Their program's topological sort will be tested against the two sample files that they were given.

- 4 points: Doxygen commenting
  - Although the middleearth.h/cpp files are not required for the code to compile, they are required to be doxygenated, along with topological.cpp
  - 2 points is given for proper commenting (0 points for no commenting, 1 point for the absolute minimal commenting possible, and 2 points for well-done commenting)
  - 2 points is given for using an additional doxygen tag beyond what was provided in the tutorial (and listed above)
  - Thus, if they did not use any additional tags, they can get at most 2/4 for this part
  - -1 to this part if their code does not create the doxygen output via 'make doc'
- 6 points: Proper topological sort
  - 3 points for each correct output test run (there were 2 test runs)

Pre-lab input

There were two files provided for the input: prelab-test-small.txt and prelab-test-full.txt, but 4 execution runs (see below).

```
cs2110 cs2150
cs2102 cs2150
cs1110 cs2110
cs3330 cs4414
cs2150 cs4414
cs2110 cs3330
cs1110 cs2102
==========================================================
cs1110 cs2102
cs1110 cs2110
cs2102 cs3102
cs2110 cs2150
cs2102 cs2150
cs2110 cs3102
cs2102 cs4102
cs2150 cs4102
cs2150 cs3240
cs2150 cs4414
cs2330 cs3330
cs2110 cs3330
cs2110 cs2190
cs3330 cs4414
cs2110 cs3205
cs3240 cs4620
cs3330 cs4620
cs4810 cs4830
cs2150 cs4610
cs2150 cs4710
```

```
cs2150 cs4240
cs2150 cs4753
cs2150 cs4750
cs2150 cs4810
cs3330 cs4330
cs3330 cs4444
cs3330 ece435
cs3330 cs4434
cs3330 cs4457
cs4457 cs4458
cs2150 cs4330
cs2150 cs4444
ece435 ece436
cs2150 cs4630
```

Pre-lab output

For the first test run (prelab-test-small.txt), there are five valid topological sorts; any one of them is valid for credit.

- cs1110 cs2110 cs2102 cs3330 cs2150 cs4414
- cs1110 cs2110 cs2102 cs2150 cs3330 cs4414
- cs1110 cs2102 cs2110 cs3330 cs2150 cs4414
- cs1110 cs2102 cs2110 cs2150 cs3330 cs4414
- cs1110 cs2110 cs3330 cs2102 cs2150 cs4414

For the second test run (prelab-test-full.txt), there are many topological sorts. The input file shows the CS course pre-requisite graph, which can be seen here. Any valid topological sort is fine for credit.

The third test run attempts to find the old doxygen tags that they were supposed to use (@brief, @return, @param, @todo), but this will only work if they used the @version, not the \version. So if nothing is found, you still have to look into their code.

The fourth attempt attemps to find new doxygen tags that they were supposed to add - the same caveat as above applies (only if they used the @version, etc.).

Note that if they have a space between the '@' and the doxygen tag, then it will appear in the fourth execution run, not the third.

In-lab

Out of 10 points. This lab part **is** expected to compile.

For the doxygen commenting, they should have something beyond the tags that were in the lab PDF (those tags were @brief, @return, @param, @todo). This will require briefly looking at their code.

- 4 points: Doxygen commenting
  - All three of the submitted source code files (traveling,cpp, middleearth.h/cpp) should be properly doxygenated.
  - 2 points is given for proper commenting (0 points for no commenting, 1 point for the absolute minimal commenting possible, and 2 points for well-done commenting)
  - 2 points is given for using an additional doxygen tag beyond what was provided in the tutorial (and listed above)
  - Thus, if they did not use any additional tags, they can get at most 2/4 for this part
  - -1 to this part if their code does not create the doxygen output via 'make doc'
- 6 points: Proper traveling salesperson problem
  - 2 points for each correct output test run (there were 3 test runs)
  - If they did the algorithm correctly, but somehow got the city list incorrectly (used the wrong random seed, started at the wrong part of the cycle, etc.), then take off 2 points (out of 6) for that type of error

In-lab input

The traveling salesperson problem was run with three different inputs. Since it is running on the same machine, the generated world (and thus the shortest path) should be the same. The three test runs had a 20x20 world, with 20 cities, a random seed of 10, 11, and 12 (respectively), and 7 cities to visit.

In-lab output

Keep in mind that if the cycle is reversed, it's still a correct solution - both directions are shown below. All three test runs had a 20x20

world, with 20 cities, a random seed of 10, 11, and 12 (respectively), and 7 cities to visit.

- Execution run for random seed 10: minimum cycle has length 60.0479
  - Minimum path has distance 60.0479: Pelennor Fields -> Isengard -> Osgiliath -> Fangorn Forest -> Erebor -> Helm's Deep -> Bywater -> Weathertop -> Pelennor Fields
  - Minimum path has distance 60.0479: Pelennor Fields -> Weathertop -> Bywater -> Helm's Deep -> Erebor -> Fangorn Forest -> Osgiliath -> Isengard -> Pelennor Fields
- Execution run for random seed 11: minimum cycle has length 51.5787
  - Minimum path has distance 51.5787: Orodruin -> Bywater -> Mirkwood -> Moria -> The Grey Havens -> Osgiliath -> Entwash River -> Edoras -> Orodruin
  - Minimum path has distance 51.5787: Orodruin -> Edoras -> Entwash River -> Osgiliath -> The Grey Havens -> Moria -> Mirkwood -> Bywater -> Orodruin
- Execution run for random seed 12: minimum cycle has length 56.1789
  - Minimum path has distance 56.1789: Orodruin -> Minas Morgul -> Pelennor Fields -> Isengard -> Emyn Muil -> Barad-Dur -> Rivendell -> Trollshaws -> Orodruin
  - Minimum path has distance 56.1789: Orodruin -> Trollshaws -> Rivendell -> Barad-Dur -> Emyn Muil -> Isengard -> Pelennor Fields -> Minas Morgul -> Orodruin

---

Post-lab

Out of 10 points. This lab part is **not** expected to compile.

The explanation of the time and space analysis is worth 4 points, and each of the 3 required acceleration techniques are worth 2 points each.

- 10 points: inlab10.pdf
  - -10 for nothing
  - -4 points max for incomplete explanation of time and space analysis (-1 for time and space for in-lab and pre-lab)
  - -2 points for missing one of the following requirements in the report:
    - Acceleration Technique #1
    - Acceleration Technique #2
    - Acceleration Technique #3
  - Grading Specifics for Acceleration Techniques:
    - -.50 points overall for not including any technique runtimes
    - -.50 points overall for not including how much faster it would make the in-lab code
    - For missing any of the following requirements per acceleration technique:
    - -1 point for poor explanation or explanation that is not detailed enough
    - -1 point if a technique's explanation was very obviously not researched or only a paraphrase from Wikipedia (all sites linked to the Traveling_salesman_problem webpage) and/or the top 2 hits in Google for "traveling salesman problem". In other words, the report did not add anything new to what was said in any of the following sites:
      1. http://en.wikipedia.org/wiki/Traveling_salesman_problem
      2. http://en.wikipedia.org/wiki/Branch_and_bound
      3. http://en.wikipedia.org/wiki/Linear_programming
      4. http://en.wikipedia.org/wiki/Cutting-plane_method
      5. http://en.wikipedia.org/wiki/Genetic_algorithm
      6. http://en.wikipedia.org/wiki/Tabu_search
      7. http://en.wikipedia.org/wiki/Dynamic_programming
      8. http://en.wikipedia.org/wiki/Ant_colony_algorithm
      9. and possibly other wikipedia entries…
      10. http://www.tsp.gatech.edu/
    - -2 points for each acceleration technique completely missing

---

Untitled note

Find a notebook

Add tag

Add comments

Version 6.0.6: c128369/1.0.1.250

Web Clipper tutorial

Options

Saving clip...

To:

Share

Save

Selection

PDF