CS 251: Code Warrior: gdb, Profiling, doxygen, Pyplot: Inlab

- Due: 8:10 PM 27/09
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on Piazza.

Overview

As a code warrior, it is inevitable that you will (legally) use some one else's code, as you did for Box2D. However what if the code has a bug, or doesn't work as well as you want to? Enter gdb from the GNU project, and gprof.

And how do you enable others to use YOUR code? That's where documentation comes into the picture, and doxygen is a workhorse. doxygen generated documentation is a pleasure to use.

Sometimes the right tool for the job saves you manhours. You have seen how to make plots in Octave, gnuplot, and in R (sort of). While each has its strength, pyplot adds to your code warrior repertoire.

A. Doxygen

Doxygen and many of its variants are commonly used tools for generating clean documentation just from source code. Doxygen makes uses of a special type of comment blocks in order to recognise comments that are meant for it. Read up on special comment blocks and more from here: http://www.stack.nl/~dimitri/doxygen/manual/doxygen_usage.html

1. Download a copy of the box2d code and **document** the file
   - `cs251_base_code/src/render.hpp`
2. You will get an extra 2 marks if you can come up with a doxy comment that will leave the TA grading the lab in a fit of laughter (only technical comments please) [Mention ONLY the line number in the `Readme.txt`]
3. Next generate the doxygen configuration file.
4. In the configuration file change the project name to yours, and give a brief description to the project
5. Also change the variable FILE_PATTERNS so that it matches only the type of files present in `cs251_base_code/src`
6. Finally since we dont want to do anything with LaTeX make changes to the configuration file so that it does not generate any LaTeX output
7. Give your project a logo and make the required changes in the Doxygen configuration file
8. Run doxygen to generate the HTML output (use the makefile)

Submit the files which you've documented and the doxygen configuration file.

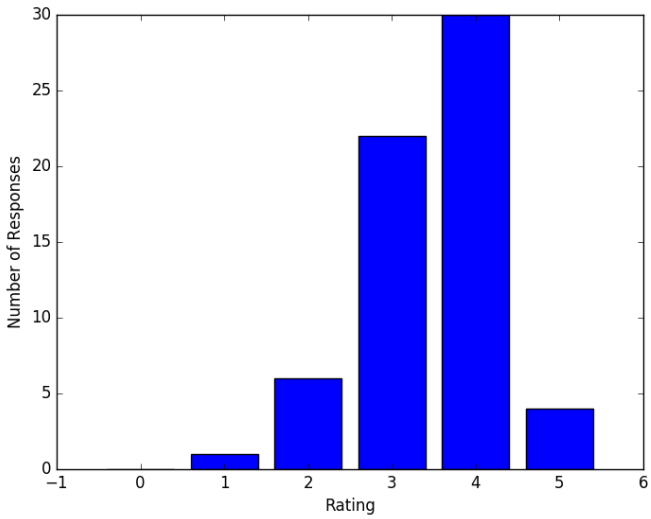
B. GDB

1. Download the executable file `prog` and the file `function.cpp`. The executable was generated using two cpp files `main.cpp` and `function.cpp`, but we have given you only `function.cpp`.
2. Run the executable. You'll notice that it takes a very very long time to terminate. We will figure out why. Inspect `function.cpp` to figure out why. Hmmn.. doesn't look suspicious.
3. Open the executable with gdb, and set a break point at the start of `calc`, and run the program. GDB stops just before executing the first line of `calc`.
4. Continue execution using the `continue` command. GDB again stops at the same break point when the function is called again.
5. Step to the next line of the code and print the `decimal` value of `minVal`
6. Add a break point at line 11 of function.cpp.
7. Once you are at line 11 print the value of `minVal`, `A.size()` and `A.size() < minVal`
8. Next find the datatypes of `minVal` and `A.size()`
9. Mention the reason as to why it takes very long for the program to terminate.

For A.3, A.5, A.6, A.7, A.8 and A.9 mention the commands used and along with reasoning in the `gdb.txt` file

C. Pyplot

1. Download the post midsemester feedback in TSV format from https://docs.google.com/spreadsheets/d/1A2xttgmr2N3L4K9Nqtnn9HQugr4ThiD-wqPtMgPTueA/edit#gid=1567016953
2. If you look at the analytics of the feedback i.e. https://docs.google.com/forms/d/14mjjwn_6w2h3NdGIdDJuSq8xli-SIcvYYjCtxfefY5A/viewanalytics you'll notice that a substantially large number of the respondents feel the course is "very good" (so far).
3. We are interested in what this majority set of people feel feel about each of the 6 labs. Again, these are only the people who felt the course was "very good"
4. Write a python script `plot.py` that reads the `tsv` file of the feedback and generates 6 bar graphs corresponding to the ratings of each of the labs (again, when taking into account only the people who rated the course as "very good").
5. The plot below is for the first lab.



Is there any anomaly with how the people feel overall, and some lab X? You can mention your observations in `Readme.txt`

D. Profiling

1. Download the input files `input1.txt` and `input2.txt`.
2. Download the source files `bubbleSort.cpp` and `selectionSort.cpp` and the executable `mergeSort`.
3. Use the executable `mergeSort` and source files `selectionSort.cpp` and `bubbleSort.cpp` to sort the data in both the input files. (use input redirection for the executable).
4. The format of the input files is as follows
   - First line contains N - the number of input values
   - Next N lines contain the values to be sorted
5. Compare the performance of the three algorithms on each input file using `gprof` or `time` and mention the average running time of the algorithms on each data set in `Readme.txt`
6. Draw the call graph for the program `mergeSort` using Inkscape. Use `gprof` to figure out the call graph and ignore functions with their names starting with '_'. Submit the image of the call graph as `taskD.png`.
7. Also include in `Readme.txt`, the sequence of commands used to figure out the call graph of Merge sort.
8. **Extra Credit :** Use the STL function sort() to sort input file `input1.txt` and draw a call graph for the sort() function in `extraTaskD.png` using Inkscape.

Submission Guidelines

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10%. In this example, the second student will get full marks (10/10) and the first student will receive 8/10.
2. Do include a Readme.txt (telling me whatever you want to tell me). (The reflection essay is not required for inlab, but is required for outlab.) Do include group members (name, roll number), group number, honour code, citations etc.

3. The folder and its compressed version should both be named `lab08_groupXY_inlab` for example folder should be named `lab08_group07_inlab` and the related `tar.gz` should be named `lab08_group07_inlab.tar.gz`

```
lab08_groupXY_inlab
├── render.hpp
├── Doxyfile
├── projectLogo
├── gdb.txt .2 plot.py
├── taskD.png
├── extraTaskD.png (extra)
└── Readme.txt
```

How We will Grade You - 30 + 8 Marks

1. Task A - 6 + 2 Marks
   (a) A.1 Documenting the files - 2 Marks
   (b) A.2 Doxygen comment - 2 Marks [Extra Credit]
   (c) A.4 Project name and description - 1 Marks
   (d) A.5 Correct FILE_PATTERNS - 1 Marks
   (e) A.6 No Latex output - 1 Marks
   (f) A.7 Project logo - 1 Marks
2. Task B - 9 Marks
   (a) B.1 Setting the break points - 1 Marks
   (b) B.5 Stepping to the next line of code - 1 Marks
   (c) B.6 Break point at line 11 of `function.cpp` - 1 Marks
   (d) B.7 Printing the 3 values - 2 Marks
   (e) B.8 The two data types - 1 Marks
   (f) B.9 Reason - 3 Marks
3. Task C - 7 Marks
   (a) Reading the data - 1 Marks
   (b) Storing the data in appropriate variable - 2 Marks
   (c) Generating the plot - 4 Marks
4. Task D - 8 + 6 Marks
   (a) Comparing the performance - 2 Marks
   (b) Average running time for all the algorithms - 1 Marks
   (c) Call graph for Merge sort - 4 Marks
   (d) Commands used for generating the call graph - 1 Marks
   (e) Call Graph of sort() function - 6 Marks [Extra Credit]