CS 251: [Code Warrior] git and python: Inlab

- Handed out: 9/21 2PM. Due: 9/21 4:45PM
- 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 the web page.

Overview

The days of individually writing useful software programs is over. It's all to be done in a team. The goal of this lab is to introduce version management using **git**.

This lab will also give you a basic introduction to **python** as file manipulation scripting language.

Tasks

1. **[Git]** The instructions in this lab are elaborate. For best results, read the entire document before proceeding. Important: This lab must be completed by the group logging in the computer as `userHighest`. Read ahead. [40 marks]

   (a) Collect all the changes you have made in your Box2D implementation in an area which is owned by the middle roll number in your group. (This could be at least Lab 03 in case you did not make any development on YOUR project). If there are only two members in your group, the middle roll number is that of the lowest roll number.

   (b) Collect all the documentation related to your updates and place them in an area owned by the lowest roll number in your group.

   (c) The original Box2D source code is supposed to have some bug in their timing code, so we are going to apply two patches to it. Download the two patch files: `b2Timer_cpp.patch` and `b2Timer_h.patch` [5 marks]

   - Make a copy of the files `b2Timer.cpp` and `b2Timer.h` from the `external` directory. The destination is `/tmp/userHighest`
   - Apply the patch to both the files. (How?)
   - Find the difference between the patched versions and the originals you copied and make sure that the patch is applied.
   - Write down your observations in the readme file. How many lines were changed?

   (d) Try out `git` at http://try.github.com/ if you have not already. Make sure you read the "advice" at the bottom and the instruction in the top. [0 mark]

   (e) We are now going to make an updated Box2D distribution which is going to contain your code and your documentation. Start with creating a local directory called `tempHighest`, and then a local (master) bare git respository called `git_cs251_highest` which contains no data.

   Stage and commit all the files in the original Box2D ~~repository~~ distribution (corpus) that was provided to you with the patch applied.

   Note that the repository should contain only the source files, and Makefile. No binaries or Object files should be there. This means you should add to the repository only

after running `make clean` and `make dist clean`. Also, while committing changes to the repository in later steps (see below), make sure that you are not adding binaries or object files in it. [15 marks]

Whenever you commit, add meaningful notes. All commits must contain this line.

```
This change is performed by "your name" of Group "your group name" at
‘date‘".
```

(f) Start a new `xterm` or terminal session. User `lowest` will ssh into his (or her) `public_html` area, create a directory called lab07 and now create a remote repository. The repository must be accessible at `www.cse.iitb.ac.in/~username/lab07/git_cs251_gXX`. There should be only one remote repository and should be present in the `public_html` folder of the group member submitting the assignment. [10 marks]

(g) Switch back to user `highest` and then push all the files into the remote repository created by `lowest` [5 marks]

(h) Log in as `userMiddle`. Clone the remote directory into an area called myproject, copy your files from your pendrive to this area, and push the changes to the remote repository. [We are using pendrive symbollically, it could be from any other area we are not aware of.] [5 marks]

(i) Switch back to user `highest` and pull the changes and make sure that you have your latest Box2D code!! [5 marks]

(j) Repeat steps so that all the documentation collected by the lowest roll number is also in the repository

2. [**python**] Suppose you want to build a utility which identifies **comments**, in the source code of computer programs.

   Programs are written either in C, C++ or Java. We are handling single line comments `//` and multiline comments using `/*` and `*/`. We will not handle any kind of nested comments.

   All possibilities that you need to consider are available in the data folder. You can use the program **test_code.cpp** and the required output file **test_code_out.txt** to test your code.

   Your task is to write a `python` program, which accepts a C program and outputs only the contents of the comments. [10 marks]

Submission Guidelines:

Submit the following documents:

1. **Task 1:** Submit the `git log` output displaying all the changes that you had performed in remote repository as `log.txt` with a `readme.txt` file. Also make sure you submit the remote repository. Write all your observations, and the objective of this part.

2. **Task 2:** `comment_finder.py`

Do not forget to put `readme.txt` file in a folder. The folder and its compressed version should both be named `lab07_groupXY_final`. Hence, you submit a tar.gz named `lab06_group07_final.tar.gz` if your group number is 7.

How We will Grade You

- Honor Code and package complete in all respects +2. **Incorrect or incomplete -2**.
- Marks corresponding to all the question is given along with the question itself.
- Code for question 2 will be tested on the given input file as it has all the different test cases.