COMP2203 Lab 1: Basic Git Version Control

Objectives

By the end of this lab, you will have:

- Forked (copied) an existing Git repository.
- Cloned the contents of a Git repository into a working directory.
- Identified and remedied bugs in the forked code.
- Committed your changes into your Git repository, and documented it in a reasonable way.

Grades

- You must record the preparation points in your logbook for a D.
- To get a C, additionally complete part 1.
- To get a B, additionally complete part 2.
- To get an A, additionally complete part 3.

When recording answers in your log book, write the section and question number. **Write clearly and legibly** – scruffy work will lose you marks.

Your mark will improve if you use the correct tools, and write readable, commented code. We are happy to help you find the right tools and coding style.

Preparation

To get started, you will need a Github account. We will assume you don't have one.

a. Browse to www.github.com.

b. Sign up for a free account.
   Github is a hosting service for Git repositories, which also features many useful tools for managing your development project.
   With a free account, you can have an unlimited number of publically-viewable repositories.
   Tip: If you sign up with your University email address, Github will give you five free private repositories (i.e. not publically-viewable).
   Head over to www.github.com/edu after signing up to claim your free repositories.

c. Install the Github client on your workstation. It can be found under "Additional Software", from the Desktop.

Part 1: Forking a Repository

Boyd's Used Motors is a trusted retailer of "nearly new" automobiles. Unfortunately, the company has a particularly poor website. Over the coming weeks, it will be your task to transform the website into a modern-looking PHP and database-driven site.

The current website contains many bugs, and it will be your job to fix them in this lab.

Conveniently, Boyd has committed all the code for his website into a Github repository, so you can take a copy and begin making improvements.

a. Browse to the website repository, at www.github.com/COMP2203/lab1

b. Fork the repository into your new Github account, so that you have you own copy to work with. Give the new repository a sensible name.

c. Create an empty directory in your ECS webspace called comp2203-lab1.

d. Use the Github Client application on the Lab PCs to connect to Github and Clone your newly-forked repository into your newly-created directory.
The Clone is your working copy, and will be linked to the master repository at Github.

e. Check that the site is running properly in your webspace, by visiting http://users.ecs.soton.ac.uk/[you]/comp2203-lab1.

Part 2: Correcting Bugs

In this section, you will fix several bugs in the Boyd's Used Motors website, and learn how to commit these changes into a Git version control system.

The Git application automatically detects when changes are made to code in your working directory, and will offer to Commit these changes to the Repository.

We recommend working within the following rules:

- *Address a single issue or feature within each Commit.*
  Doing "one thing at a time" with your Commits makes it easier to track changes and analyse how the project has developed. Colleagues will also find it easier to understand what you've been working on.

- *Write a Commit message every time, which is concise and meaningful.*
  The Commit message is broken down into a Subject and Detailed section, much like an email. Always write a brief Subject that describes the changes in your Commit.

- *Commit working code whenever possible.*
  It may be tempting to "save" your work by Committing a half-implemented feature, or a half-fixed bug.

Ten JavaScript bugs have been identified within the website. These bugs are a mix of syntax bugs, logical bugs and conceptual errors.

While plenty of other problems exist within the site, you will only receive marks for resolving JavaScript-related errors. We will address the other issues in future labs.

For each bug you find…

a. Resolve the bug and test the website.

b. Commit a change into your repository, documenting the bug and how you resolved it.

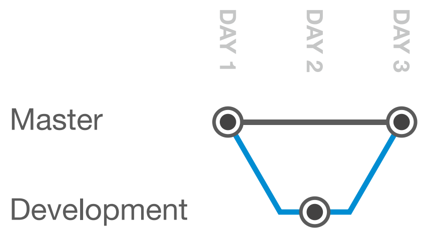
c. Push your updated repository back to the master repository on Github.

Part 3: Understanding Commit Graphs

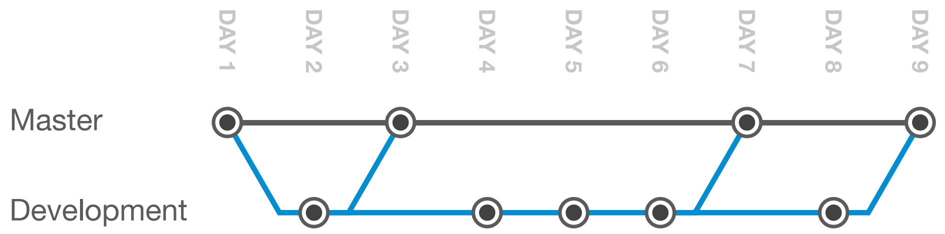
Commits to a Git repository can be shown on a graph, making it easier to understand how the code has evolved over time. This is particularly useful when working with other people, and on complex projects.

In this section, we'd like you to write a concise description of what is happening in each of the following commit graphs.

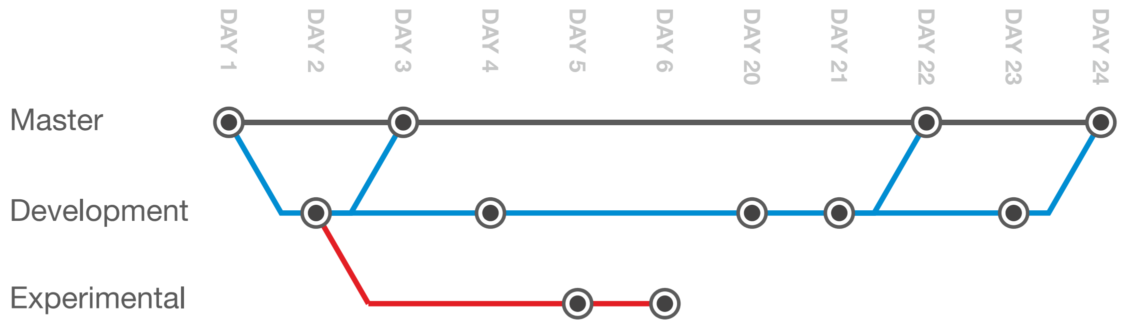
a.



b.



c.



Now take your log book to your demonstrator for marking.

Your Github settings should remain on the workstation you used this week.

So, if you use the same workstation next week, you may save some time.