Continuous Integration

Group 30 Triple 10

Team Members:
Kelvin Chen, Amy Cross, Amber Gange, Robin Graham, Riko Puusepp, Labib Zabaneh

Part a)

We have adopted a continuous integration methodology of pushing as much as we can as often as we can, no matter how small the change is. This has allowed us to keep on top of what everyone is doing and ensure that we have a stable build for every change we make.

Our project is a fairly small and robust build that doesn't necessarily require all the advanced automated building features you might see for larger scale projects. This has allowed us to approach our continuous integration by manually having to push and pull commits to our repository. This would involve running only a few lines of code in order to keep up to date with the latest version of the project. We also had to build automated tests in order to run part of the build in order to check that the build will work. This allows us to ensure that the project can be safely built for anyone to download which saves us time having to debug codes we didn't see on commits on the branch before. We would then approach these problems by fixing them immediately so other group members can work on the project safely.

Another methodology we appointed was to make sure to use the least amount of branches as possible to ensure we're all working on the same pipeline workflow as seen on GitHub actions.

Essentially, our approach is designed to be fast paced as it's quick to pull any update, it's robust by running automated tests and using Github actions to keep track of when and what people have updated into the repository.

Part b)

Our pipeline continuous integration approach for our small group based project is described below.

As mentioned above, we decided to use GitHub and Git to continuously implement our updates onto our repository and we would get notifications when updates are implemented and when. We decided on the project to have 2 branches, one main branch and one testing branch. This allowed us to keep our testing separate from the main source code allowing for safe testing with a stable build always being available when needed.

Our Continuous Integration Process

First we would make sure our project is connected to our repository then open a Git bash terminal to run some of the following commands.

1) Git add -A
    - Update index of current content found in the branch to prepare for any commits
2) Git commit -m "message"
    - We then commit our code into our local environment which we can push onto the main repository if updated

3) Git pull
    - We then pull the latest version from the GitHub repository onto our systems to ensure that we have the latest version to work on and that everyone
4) Git switch <branch>
    - Here we switch to the appropriate branch for our push to be committed to. We only have two branches so this is fairly self explanatory
5) Git push -m "message"
    - Finally, we need to push the updated code onto our code repository taking great care to write appropriate messages to describe what the push was
    - Since we're doing little pushes as often as possible this won't be extremely detailed for each individual push

With this, we could smoothly work on our project by always adding our new elements on top of a stable and working build which allowed for fast continuous integration leading to an efficient and effective working environment where anyone could make changes without having to fear breaking the entire build due to suitable testing we have employed.