IT231 Lab 1
Score: ______/17 v2.1

In this lab, you'll get some experience with Git commands and merge conflicts.

Specifications

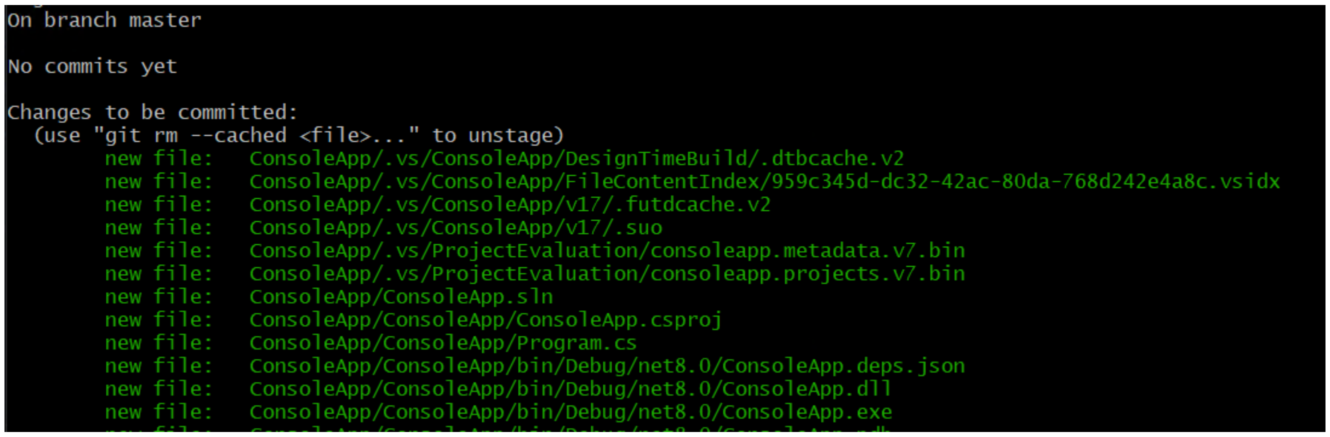
Perform all tasks with numbered points using the CLI. Save the commands for the submission, and ensure each step is clearly identified with a preceding number indicating the corresponding command or commands.

- Ensure Git is installed.
  - https://git-scm.com/downloads
- Create a repository in GitHub.
  - Name the repository "IT231-Lab1".
  - Ensure the repository is public.
- (Windows only) Open the Git Bash terminal. Search for it as an application.
- (Mac and Linux only) Open the terminal.
- Create a directory on your computer to clone to.

1. Clone the repository to the directory you made for the lab.
   a. Ignore any warnings about cloning an empty directory.
2. Change your directory to be the directory you cloned.
   a. You should see it says "master" next to the filepath.

- Create a C# console app program in the directory you cloned. Don't modify the project. Build the project. Close your IDE.

3. List the files you've changed.
   a. It should show no files. Consider why that is.
4. Start tracking all of the files in the directory.
   a. The keyword is not "track".
   b. Use a Git command to ensure that all of the items are being tracked. You should see something like this:



```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   ConsoleApp/.vs/ConsoleApp/DesignTimeBuild/.dtbcache.v2
        new file:   ConsoleApp/.vs/ConsoleApp/FileContentIndex/959c345d-dc32-42ac-80da-768d242e4a8c.vsidx
        new file:   ConsoleApp/.vs/ConsoleApp/v17/.futdcache.v2
        new file:   ConsoleApp/.vs/ConsoleApp/v17/.suo
        new file:   ConsoleApp/.vs/ProjectEvaluation/consoleapp.metadata.v7.bin
        new file:   ConsoleApp/.vs/ProjectEvaluation/consoleapp.projects.v7.bin
        new file:   ConsoleApp/ConsoleApp.sln
        new file:   ConsoleApp/ConsoleApp/ConsoleApp.csproj
        new file:   ConsoleApp/ConsoleApp/Program.cs
        new file:   ConsoleApp/ConsoleApp/bin/Debug/net8.0/ConsoleApp.deps.json
        new file:   ConsoleApp/ConsoleApp/bin/Debug/net8.0/ConsoleApp.dll
        new file:   ConsoleApp/ConsoleApp/bin/Debug/net8.0/ConsoleApp.exe
```

Often, you do not want to add all of the files to your repository. Really, in a C# project, the only useful files are the source files, the project files, and the solution file.

- Create a gitignore file and ignore the .vs, bin, and obj directories.

5. Add the gitignore file and ensure it is working.
    a. If all of the files have already been added, you will need to remove them (or all of the files) and re-add them.
    b. Do not delete any files.
    c. All commands need to be shown in your submission.
    d. You may use Google or AI for help if needed for this step.
6. Commit your project.
7. Push your project to GitHub.
    a. Check your GitHub repository on github.com. Make sure your files are there.

- Make a modification to your code. Take two integer inputs from the user via the console. Save the inputs in variables. Save your code.

8. Push your changes to GitHub.
    a. Check your GitHub repository on github.com. Make sure your change is there.
9. Create a branch called “math” and switch to it.

- Make a modification to your code. Create a method called “`DoMath()`” that has two parameters, *multiplies* the numbers, and prints the result. Call the method in the `Main()`, pass in the two input variables. Save your code.

10. Commit the changes, but do **not** push your changes.
11. Switch back to the master branch.

- Make a modification to your code. Create a method called “`DoMath()`” that has two parameters, *adds* the numbers, and prints the result. Call the method in the `Main()`, pass in the two input variables. Save your code.
    a. Essentially, it should be the same code you wrote for step 13 except it adds instead of multiplying.

12. Commit and push your changes.
    a. Check your GitHub repository on github.com. Make sure your change is there.
13. Switch back to the “math” branch.
14. Push those changes.
    a. Check your GitHub repository on github.com. Make sure your branch and change is there.
15. Merge the “math” branch into the master branch.
    a. This should fail.
16. Let’s assume that neither the addition nor multiplication version is incorrect, and in fact, you want to do both. Resolve the merge conflict in such a way that your code now multiplies the two numbers, adds the two numbers, and prints the results from both calculations.

    a. Write down everything you do to resolve the merge conflict, including non Git commands.
    b. Make sure your merge appears in your GitHub repository.
17. Delete the multiplication branch. Ensure it is deleted from the remote GitHub repository.

Submission

The submission for this lab are the commands used during the lab. You’ll also explain how you fixed the merge conflict. Lastly, submit the your GitHub repository link.