Directions for CS455: Introduction to High Performance Computing

Standard Development and Submission

Below are the requirements for all assignments, which are standard. In all cases, the local directions for assignments take priority over those specified here.

1. **Initial Setup:** Create a `development` branch in your repository. Perform all your work in this branch.
2. **Regular Commits:** Make commits to your local repository regularly and push to GitHub at key milestones for backup.
3. **Submission Deadline:** Refer to the class schedule for deadlines. If not specified, the **default** due time is 11:59 PM the day the assignment is due. Submit your assignment by initiating a *pull request*. **DO NOT merge** with the `main`; the grading process will include that step.
4. **Late Submissions:** See the syllabus for the Late Work Policy.

Requirements for All Submissions

These items **required for all assignments**, even if they are not specified in the assignments README.

Use of `development` Branch

After accepting your assignment, your first task is to create a `development` branch in your repository. The simplest way to create this branch is through the browser. You **MUST** complete all your work within the `development` branch. This requirement means that every *commit* you make and every *push* you perform must be directed specifically to the `development` branch.

All Files Changed Modified Doc-Box

Any source code file you modify (i.e., those ending in .cc) requires the following Doc-Box added.

```
/*
    <Your Name>
    <Assignment>
    <Date>

    I certify that this is my work and, where appropriate, an extension of
    the starter code provided for the assignment.
*/
```

Makefile Requirements

Add a *Makefile* to your repository unless otherwise specified. All Makefiles should include three essential targets: the program (e.g., `a01`), `make all`, and `make clean`. The target for the program (e.g., `a01`) should compile the necessary source files and generate the executable.

The `make all` target should be a convenient way to build the entire project, ensuring all dependencies are compiled correctly. It should invoke the program's compilation target so that running `make all` produces the final executable(s) without requiring manual specification.

The `make clean` target should remove all compiled artifacts, including object files (`.o`), executables, and other generated files. This ensures a clean working directory, allowing for a fresh rebuild without leftover

files from previous compilations. Running `make clean` should not produce errors, even if there are no files to remove.

Reflection Requirements

All assignments require a reflection paragraph unless specified otherwise. Name the reflection file after the assignment (e.g., `a00.txt` for assignment *a00*). Ensure it is added to the repository, even if `.gitignore` may require a forced add. The reflection must be **well-written**, with **proper grammar and spelling**, and should provide meaningful insight into your experience with the assignment. Consider discussing your initial thoughts, key takeaways, challenges faced, unexpected difficulties or discoveries, and any strategies you used to overcome obstacles. Additionally, reflect on how the assignment contributed to your learning and suggest potential improvements for future iterations. Thoughtful and detailed reflections will help you better understand the material while providing valuable feedback for course improvements.

Documentation and Code Quality

The following guidelines must be followed for assignments developed in this course.

1. Include the required doc-box in all files you write or modify. Omission will affect your grade.
2. Follow a standard approach to documentation. For example, the Google C++ Style Guide.

Additional Resources

1. **Git Commands Cheat Sheet:** A quick reference to essential Git commands. It helps manage your code and changes efficiently.
2. **Learning Git and GitHub:** A collection of resources curated by GitHub to help you understand how to use Git and GitHub effectively for version control and collaboration.
3. **Makefile:** A very nice guide to the making of Makefiles.