Проект по параллельному программированию

1. Используется соглашение о коде:

- используется верблюжий стиль для имен, классы начинаются с большой буквы (RacingCar), функции – с малой (doItYourself()).
- абстрактные классы начинаются с заглавной буквы I (ICar).
- имена классов, функций, полей, методов – английские слова и словосочетания, отражающие их смысл, или их сокращения (метод класса Машина Car движение вправо – moveRight())

Используется язык C++, либо C#, либо Java. В части 1 используются объекты-потоки (C++ Threads), в части 2 – автоматическое распараллеливание циклов (OpenMP), в части 3 – строго реализация MPI.

Используется git, причем

а) один коммит – одно логическое изменение (!). Измените привычку, и делайте БОЛЬШЕ коммитов. Ваш стиль разработки будет оцениваться в соответствии с этими правилами.

б) каждый коммит в первой строке сообщения кратко описывает изменения (напр.: добавил функцию вычисления факториала). Если нужно, в последующих строках делается детальное описание смысла изменений.

Преподаватель просматривает историю коммитов при защите части проекта, оценивает число коммитов, их содержательность, количество работы, выполненное каждым членом команды. Оценивает выполнение каждым членом проекта своих задач. Это вместе с правильностью решения поставленной задачи (корректность решения, потокобезопасность, оптимальность выбранного метода, чистота кода (следование соглашению о коде, наличие комментариев, использование современных средств (вектор предпочтителен массиву в стиле Си и т.п.)) формирует итоговую оценку за часть проекта.

2. Проект выполняется в командах по 3 человека. Все должны иметь аккаунты на Github, которые будут известны преподавателю и членам команды (если Вы хотите оставить свой основной аккаунт анонимным, создайте для учебных целей отдельный аккаунт)
3. Проект состоит из трех частей: Threads, OpenMP, MPI. При выполнении каждой части все члены команды получают свои роли. При выполнении следующей части члены команды меняются ролями. Таким образом, за три части проекта каждый должен попробовать все роли по одному разу. Есть три роли: разработчик (кодер), тестировщик (тестер) и руководитель-мэйнтейнер+DevOps (DevOps). Запомните, что часть 3 – MPI – требует наиболее сильного кодера (и тестировщика).

4. DevOps создает ПРИВАТНЫЙ репозиторий и помимо ветки main ВЕТКИ для тестера и кодера. Дает доступ к репозитории членам команды (на редактирование) и преподавателю (по крайней мере на комментирование, возможно - на редактирование).
5. **ЗАДАЧИ кодера:** Ведет разработку по решению задачи в своей ветке. Делает коммиты (правила в пункте 1). Мерджится и активно работает с тестером. В коде делаются комментарии.
6. **ЗАДАЧИ тестера: Регулярно обновляется с ветки кодера и мерджится с кодером**. Ведет написание тестов (дополняя файлы кодера или в отдельных файлах) в своей ветке. Делает коммиты (правила в пункте 1). Мерджится и активно работает с кодером. И делает пулл реквест в мэйн, когда закончил, оттестировал сам, и думает, что все работает, задача решена и надо протестить ее в Github Actions.
7. **ЗАДАЧИ DevOps:** Настраивает Github Actions для автоматического построения проекта и запуска с тестами. Проверяет код и принимает пулл реквест (мерджит), либо не принимает (комментирует на гихаб пулл реквест, что доделать). Активно взаимодействует по срокам (пинки) с другими членами команды. Когда все сделано, и тесты проходит программы, радуется больше всех. Ему предстоит от лица команды (при их участии) защищать сделанную работу в Discord/Очно перед преподавателем для получения оценки.