Общая Цель: Создать библиотеку `ImageProcLib` для обработки изображений и консольное приложение `imgproc`, которое её использует.

Разработчик А (Фокус: Базовые операции, Сборка)

- **Что делает:** Функции для загрузки/сохранения картинок, базовую операцию свёртки, фильтр Гаусса и детектор границ Собеля. Также отвечает за настройку и поддержку `Makefile` (чтобы проект собирался).
- **Модули:** `io.c/h`, `convolution.c/h`, `edge_detection.c/h`, `filters.c/h` (часть про Гаусса).

Разработчик Б (Фокус: Алгоритмы, CLI, Документация)

- **Что делает:** Функции для преобразования в ч/б (Grayscale) и RGB, медианный фильтр, выравнивание гистограммы. Главное – создаёт консольное приложение `imgproc`, которое будет использовать *все* функции библиотеки (и свои, и от Разработчика А). Также отвечает за документацию (README, Doxygen) и настройку тестов.
- **Модули:** `color_conversion.c/h`, `histogram.c/h`, `filters.c/h` (часть про медиану), `app/main.c` (консольное приложение).

План Работы (По Фазам):

Фаза 1: Подготовка и Основы

1. **Вместе:**
   - Настроить общий Git-репозиторий.
   - Договориться об общих структурах данных (`Image`, `Kernel`), кодах ошибок и стиле написания функций (как передавать параметры, как возвращать результат).
2. **Разработчик А:**
   - Найти и подключить библиотеку `stb_image` для работы с картинками (JPG, PNG).
   - Создать базовый `Makefile` (хотя бы для компиляции отдельных файлов и очистки).
3. **Разработчик Б:**
   - Настроить Doxygen для генерации документации по коду.
   - Создать файл `README.md` и начать его заполнять (описание проекта, как собрать и т.д.).
   - Если используется фреймворк для юнит-тестов – настроить его.

Фаза 2: Ввод-Вывод и Первые Функции

1. **Разработчик А:**
   - Реализовать и протестировать загрузку (`ipl_loadImage`), сохранение (`ipl_saveImage`) и освобождение памяти (`ipl_freeImage`) для картинок.
   - **Очень важно:** Как можно скорее отдать работающий модуль `io` Разработчику Б.
   - Обновить `Makefile`, чтобы можно было собрать модуль `io`.
2. **Разработчик Б:**
   - Реализовать и протестировать функции преобразования цвета (`ipl_convertToGrayscale`, `ipl_convertToRGB`).
3. **Разработчик А:**
   - Реализовать и протестировать базовую свёртку (`ipl_convolve`).
   - Обновить `Makefile`.

Фаза 3: Основные Фильтры и Алгоритмы

1. **Разработчик А:**
   - Реализовать и протестировать фильтр Гаусса (`ipl_gaussianFilter`).
   - Реализовать и протестировать детектор границ Собеля (`ipl_sobelEdgeDetection`).
   - Продолжать обновлять `Makefile`.
2. **Разработчик Б:**
   - Реализовать и протестировать медианный фильтр (`ipl_medianFilter`). *(Нужен модуль `io` от А для тестов)*.
   - Реализовать и протестировать выравнивание гистограммы (`ipl_histogramEqualization`). * (Нужен модуль `io` от А)*.

Фаза 4: Консольное Приложение (Основная работа Б)

1. **Разработчик Б:**
   - Разработать приложение `imgproc` (`app/main.c`):
     - Сделать парсинг аргументов командной строки (чтобы понимать команды типа `-grayscale`, `-o output.jpg` и т.д.).
     - Написать логику, которая вызывает нужные функции библиотеки (от А и Б) в зависимости от команды.
     - Реализовать загрузку ядра свёртки из файла для команды `-convolve`.
     - Добавить автоматическое преобразование в ч/б перед операциями Собеля или выравнивания гистограммы, если нужно.
   - **Интегрировать** все готовые модули от Разработчика А (`io`, `convolve`, `gaussian`, `sobel`).
   - Написать и запустить интеграционные тесты для `imgproc` (проверить разные команды и их комбинации).
2. **Разработчик А:**
   - Помогать Разработчику Б с интеграцией своих модулей в `imgproc`.
   - Исправлять ошибки в своих модулях, если Б их найдёт при интеграции.
   - Дорабатывать `Makefile`, чтобы он собирал и библиотеку (`libimageproc.a`), и приложение (`imgproc`).

Фаза 5: Завершение и Полировка

1. **Вместе:**
   - Провести финальное тестирование (юнит-тесты и интеграционные).
   - Прогнать код через статические (проверка стиля, потенциальных ошибок) и динамические (проверка утечек памяти, например, Valgrind) анализаторы. Исправить найденные проблемы.
   - Исправить все оставшиеся баги.
   - При необходимости провести рефакторинг (улучшение структуры кода).
2. **Разработчик А:**
   - Финализировать `Makefile` (цели `all`, `debug`, `clean` должны работать идеально для всего проекта).
3. **Разработчик Б:**
   - Дописать `README.md`.
   - Сгенерировать финальную документацию Doxygen (`make doc`).

Ключевое для Успеха:

- **Общение:** Регулярно общайтесь, особенно на стыках задач (передача модулей, интеграция).
- **Ранняя Интеграция:** Разработчик А должен как можно раньше предоставить работающий модуль I/O Разработчику Б. Начинайте интеграцию модулей в `imgproc` по мере их готовности, не ждите самого конца.
- **Общие Стандарты:** Придерживайтесь договоренностей из Фазы 1 по стилю кода и API.
- **Тестирование:** Пишите тесты по ходу разработки, а не в самом конце.