Team Contract

Goals

- What are the goals of the team?
    - Do the project quickly (ahead of the deadline) and well enough to get an A.
- What kind of obstacles might you encounter in reaching your goals?
    - Other classes: exams, classes, or other psets might get in the way
    - Technical difficulties: code might break, or there might be conflicting commits
    - Lack of communication: somebody might go offline at an important time
- Is it acceptable for one or two team members to do more work than the others in order to get the team an A?
    - Things always come up. If we plan to get work done ahead of schedule, we will have more time to deal with these kinds of problems. We all think it's okay for somebody to do a bit more work in these situations.

Meeting Norms

- Do you have a preference for when meetings will be held?
    - Meet during lecture hour. Additional meetings may be scheduled as needed.
- How will you record and distribute the minutes and action lists produced by each meeting?
    - One google doc containing the agenda and minutes for all meetings.

Work Norms

- How much time per week do you anticipate it will take to make the project successful?
    - 10 hr a week per person
- How will work be distributed?
    - We will have a Google doc listing all of the tasks, their owners, deadlines, and whether they have been completed. https://docs.google.com/document/d/1MPSMlMEIlkBXFFr-3zxiFwQVU1X0iNWQeWM0BXZWsMo/edit
    - People's names will be placed in brackets to indicate that the task has been assigned to them.
    - To mark a task as completed, place an asterisk in front of your name in the brackets
    - Tasks will be organized into groups representing deadlines.
- How will deadlines be set & decide who should do which tasks?
    - Negotiation during meetings & over email.
- What will happen if someone does not follow through on a commitment (e.g., missing a deadline, not showing up to meetings)?
    - If the commitment is blocking the other teammates, then the workload should be redistributed to the other teammates. Otherwise, take note of the missed deadline

on the google doc, and deal with it when it becomes important.
    - We will make note of these missed deadlines in the peer review at the end of the project.
- How will the work be reviewed?
    - Code reviews will happen on Github for all commits that may break someone else's code. All tests must pass before merging into the master branch.
- What happens if people have different opinions on the quality of the work?
    - Discussion can happen on Github, and everybody on the team takes a vote. Majority wins.
- How will you deal with different work habits of individual team members (e.g., some people like to get assignments done as early as possible; others like to work under the pressure of a deadline)?
    - Deadlines and projects should be assigned to meet each person's working habits. Tasks can be swapped/redistributed to achieve this goal.

Decision Making

- Do you need consensus (100% approval of all team members) before making a decision?
    - If you are the owner of the specified part of the project, then there is no need to get a consensus before starting your work. However, if you believe that you decision will affect at least one other person's project, then you need to get approval from at least one other person on the team.
- What will you do if one of you fixates on a particular idea?
    - The project tasks should be swapped/redistributed so that nobody is designing or implementing something against the consensus of the other two team members.

Github Instructions

- Sign up for a Github account, and upload your public keys to the web interface.
- Send me (zdrach@mit.edu) your github username so I can add you to the repository
- We will use the following model
    - There will be a one repository on the github website. This repository will contain one main "master" branch which contains the stable version of our code.
    - There is a second repository on the MIT servers (the one that contained the handout code). This is where we submit our code
    - We already have the MIT repository cloned on our computers. We want to link our local repository to the new github "remote"
        - type "git remote add github git@github.com:hogbait/hairy-octo-robot.git"
    - Lets say you want to work on a new feature. If everybody is making commits to the master branch, we will be stepping on each other's feet all of the time. To sidestep this problem, you should create a new feature branch to house the stuff you're working on.
        - type "git checkout -b <feature_branch_name>
        - for example "git checkout -b parser_draft"

- Work on your changes as usual.
    - Make local commits by doing “git commit”
    - Push your commits to github by doing “git push github <feature_branch_name>”
    - Pull the latest changes from the stable master branch by typing
        - git fetch github
        - git rebase -i github/master (or “git merge github/master” if that’s more comfortable)
- You can browse your latest changes on the github web interface:
    - navigate to https://github.com/hogbait/hairy-octo-robot
    - click on the branch selector next to the green button on the left-hand side of the page. select your feature branch
    - to view the history of commits, click the “commits” button located immediately above the branch selector button
- When you want to merge your stuff into the master branch, you need to initiate the code review process
    - use the branch selector to open up your branch on the github web interface
    - click the green “compare and pull request” button that appears near the middle of the page
    - follow the instructions to send a pull request. All of our teammates should automatically receive emails whenever someone initiates a pull request. If you are not receiving emails, you probably have it disabled in your github account notification settings.
    - If you want another teammate to look over your code, mention their github username in the pull-request description by typing “@” followed by their name.
    - You can merge your stuff by click the green “merge pull request” on the pull request web page. You should only merge under the following conditions:
        - If your changes are minor, if you are fixing a bug, or if you own the particular part of the code base, you can merge right away.
        - If you are creating new code that might break someone else’s work, if you are not the owner of the particular part of the code base, or if you simply want somebody else’s approval, @ mention their github username in the pull request description and wait for their approval before merging.
    - When you are reviewing someone else’s code
        - You can comment directly on their code by clicking on the line of code in the diff
        - You can make other comments by typing in the comment box
        - Typing “ltgm” (looks good to me) to approve somebody else’s code
- Submitting the code to Didit. On your local machine, do the following.

- git checkout master
- git pull github master
- git push origin master