Current layout: 2 Panels with treePanel 0: terminal command:

**1. Writing your first C Program**

If you look at the list directory, you will see a list.h file, which will provide the functions that you need to define to implement a linked list in C. These functions should be defined in the list.c file, which is provided (but empty :).

Work with your group members to fill in these functions. What should be stack allocated? Heap allocated?

Next, write some basic tests for your implementation in the main.c file. Start off with some easy tests, and write some harder tests for your implementation later on.

You’ll also have to write your own Makefile. The Makefile will have to compile the list implementation files as well as the main.c file. Be sure that you compile your code to output an executable named list. (Hint: must use the -o flag in the compile statement).

I would highly recommend writing this a little bit at a time, to make sure your code compiles early (rather than getting hundreds of compilation errors after everything is finished). Start out with just getting the skeleton files to compile (before you write any code). Then try out list\_alloc and list\_free and see what you can do. Then do one function at a time, and test them out slowly.

Work together! It will go faster (and be less painful to debug). I will circle around to each breakout group and help. If you have a direct question, please post in zoom.

You will likely need paper+pencil/google doc to draw things. Trust me, it helps.

You probably won’t get around to implementing all the functions. That’s okay. If you didn’t complete the in class activity, finish this on your own to see how much you understand, and feel free to post any questions to slack.

**Check and Submit Your Code**

Update the file *list.c* in the terminal on the left. Then click submit to test. After you click **CHECK IT**, expand the window below to review all the results.

Current layout: 2 Panels with tree

# 2. How to use Git

# Intro to Git

Although this assignment is technically ungraded, later projects will assume familiarity with the content. Please complete the tutorial by next class.

## Step 1: Install git and create a GitHub account

Install Git

1. Open a Terminal (Go to Tools>Terminal)

If you are new to working with the command line, DON’T PANIC. It’s not as scary as it first appears.

A basic tutorial is available at <https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-18-04>

Type the following commands at the unix command line. Do not retype the $.

$ sudo apt update

$ sudo apt install git

$ git --version

If the command worked, you should see similar output

git version 2.23.0

Sign up for GitHub

1. Sign up for GitHub at <https://github.com/join>.  
   IMPORTANT: Get the “Student Developer Pack” at <http://ducation.github.com/pack>.  
   This will allow you to create a private repository for your team project.

## Step 1: Create a Local Git Repository

When creating a new project on your local machine using git, you’ll first create a new repository (or often, “repo”, for short).

1. Create (and switch to) a directory for your project. You can put it wherever you’d like.

$ cd

$ mkdir HelloGit

$ cd HelloGit

1. To initialize a git repository in the root of the folder, run the git init command:

$ git init

Initialized empty Git repository in …

## Step 2: Add a new file to the repo

1. Add a new file to the directory, using any text editor that you like.

$ echo "Hello World" > hello\_world.txt

1. After creating the new file, you can use the git status command to see which files git knows exist.

$ git status

# git status output

On branch master  
No commits yet  
Untracked files:  
Hello\_world.txt  
nothing added to commit but untracked files present

This basically says “Hey, we noticed you created a new file, but unless you use the ‘git add’ command we aren’t going to do anything with it.”

1. Instruct Git to track the file, adding it to staging.

$ git **add** hello\_world.txt

$ git status

# git status output

On branch master  
No commits yet  
Changes to be committed:  
new file: hello\_world.txt  
Once a file has been staged, Git knows to include it the next commit.

1. Commit the file to the repo.

$ git commit -m "This is my first commit to Add hello\_world.txt"

This will write a commit message (e.g., “This is my first commit to Add hello\_world.txt”). In a real project, this would summarize the purpose of the change, how it works, remaining issues, and any other relevant information.

1. At this point, your changes have been checkpointed.

$ git log

# git log output

commit 984758…5e128 (HEAD -> master)  
Author: Legand Burge  
Date: Sun Aug 26 13:01:16 2019 -0700  
This is my first commit to Add hello\_world.txt.

## Step 3: Sync to GitHub

So far, everything we’ve done was local. But since we’ll be working in teams, we’ll use GitHub to collaboratively modify the project’s code.

1. Log in to GitHub
2. Create a new repository (the big green button).  
   If you have the Student Pack, you can create a private repo.  
   Otherwise, just use a public repository for now.
3. Type the commands to “push an existing repository from the command line”

$ git remote add origin https://github.**com**/<USER>/<REPO>.git

$ git push -**u** origin master

You’ll need to enter your username and password.  
If you use 2FA, you’ll need a personal access token. See <https://help.github.com/articles/creating-a-personal-access-token-for-the-command-line/>.

1. Navigate back to your GitHub repository’s “<> Code”. You should see your file listed.

## Step 4: Sync From GitHub

1. Open your file on GitHub and click edit (the pencil icon). Change something and commit your edit.
2. Sync the changes from GitHub back to your local machine.

$ git pull origin **master**

# git pull response

1 file changed, 1 insertion(+), 1 deletion(-)

1. Notice that the history of commits has no gotten longer

$ git log

## Step 5: Working in a New Branch

In Git, development is typically done in a branch and then merged back into the main trunk (master branch) as part of a pull request. This allows a feature to be developed in isolation, without interfering with other work.

1. Create a new branch

$ git checkout -b develop\_feature\_A

# git checkout response

Switched to a new branch ‘develop\_feature\_A’

This creates a new branch and then switches you into that branch. You can see all branches via

$ git **branch**

# git branch response

* develop\_feature\_A  
  master  
  The \* indicates your current branch.  
  |||

1. Make a change to a file and add it to staging

$ echo "Hello World! This is a branch." > hello\_world.txt

$ git add hello\_world.txt

1. One quirk of Git branches is that uncommitted edits carry over to other branches.

$ git checkout **master**

**$** cat hello\_world.txt

Hello World! This is a branch.  
There is a solution to this (git stash), but for now, just be aware of it.

1. Switching back to our dev branch, commit the change.

$ git checkout develop\_feature\_A

$ git commit

1. Now that we’ve committed the edit, we can see that the branches have diverged

$ git checkout **master**

**$** cat hello\_world.txt

#output  
Hello World!

$ git checkout develop\_feature\_A

$ cat hello\_world.txt

#output  
Hello World! This is a branch.

## Step 6: Creating and Merging a Pull Request

A Pull Request is a way to alert a repo’s owners that you want to make some changes to their code. Or, in our case, to request a code review from another team member before merging a new feature into the master branch.

1. Push the branch’s current state to GitHub

$ git push origin develop\_feature\_A

1. Open your repository in GitHub and select “New Pull Request”. (There’s probably also a big green button, since we just added a new branch. That works too.)
2. We want to pull to “master” and from “develop\_featuer\_A”. Scrolling down, we can see the diff of changes. Click “Create”. At this point, you and the reviewer would discuss the commit.
3. Approve the merge. On your local client, pull the latest changes from GitHub and verify the contents of the file have been updated as expected.
4. On GitHub, go the repository’s main page. Click “Insights” and “Network”. You should see a graphical history of the repository.

## Step 7: Celebrate

You now know the basics of Git! There is lots more to master, but you can pick up the rest as you go along.

Credits: This document was adapted from  
<https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>  
<https://blog.udacity.com/2015/06/a-beginners-git-github-tutorial.html>

Mark as CompletedBack to dashboard