

CS 162 LAB #10 – Recursion & List

In order to get credit for the lab, you need to be checked off by the end of lab. Since this is the very last lab of the course, you are not able to make up points afterwards. For extenuating circumstances, contact your lab TAs and the instructor.

This lab is worth 10 points total. Here's the breakdown:

- 5 points: Implement recursion to print fractals
 - 5 points: Implement question on list and participate in discussion afterwards
-

(5 pts) Recursive Fractals

Examine this pattern of asterisks and blanks, and write a recursive function called `pattern()` that can generate patterns such as this:

`pattern(2, 2);`

```
 *
 * *
  *
```

`pattern(4, 1);`

```
 *
 * *
  *
 * * * *
   *
    * *
     *
```

`pattern(8, 0);`

```
 *
 * *
  *
 * * * *
   *
    * *
     *
 * * * * * * * *
      *
       * *
        *
         * * * *
          *
           * *
            *
             * *
              *
```

With recursive thinking, the function needs only about 10 lines of code (including two recursive calls). Your function prototype should look like this:

```
// Description:  
// The longest line of the pattern has n stars beginning in column col of the output.  
// Precondition: n is a positive even number.  
// Postcondition: A pattern based on the above example has been printed.  
void pattern(int n, int col);  
No error handling needed for this part. Assume that user will provide you a positive even int as n,  
and a non-negative int as col.
```

Hint: Think about how the pattern is a fractal. Can you find two smaller versions of the pattern within the large pattern? Here is some code that may be helpful within your function:

```
// A loop to print exactly col columns  
    for (int i = 0; i < col; i++) cout << " ";  
  
// A loop to print n asterisks, each one followed by a space:  
    for (int i = 0; i < n; i++) cout << "* ";
```

(5 pts) Question: Remove Nth Node from end of list

This question aims to help you practice CS coding questions during technical interviews using the knowledge that we've learned in the past few weeks.

To start, download the zip file from:

wget <https://classes.engr.oregonstate.edu/eecs/spring2023/cs162-010/labs/lab10.zip>

To extract it:

```
unzip lab10.zip
```

Given a linked list, remove the n -th node from the end of list and return its head.

Example:

Given linked list: 1->2->3->4->5, and $n = 2$.

After removing the second node from the end, the linked list becomes 1->2->3->5.

Note:

Given n will always be valid. (i.e. n is greater than 0)

Follow up:

Could you do this in one pass?

Hint: Maintain two pointers and update one with a delay of n steps.

Step-by-step Instructions:

1. In `lab10/list_question/`, open `solution2.cpp`
2. Type your solution there
3. Once done, run “make” to compile
4. If successfully compiled, an executable `task2` would be generated
5. Run `./task2`, and see if you can pass all 6 test cases
6. Check the time taken and memory usage of your program and compare with your classmates

Now, your Lab TAs is going to announce the “winner” of this problem in your class and let them share their solution. Join the discussion and ask questions if you have any.

Show your completed work and answers to the TAs for credit. You will not get points if you do not get checked off!

You don't need to submit anything for this lab.

That's it! You've completed all labs of the course! Congratulations! We wish you all the best in the future :)

(Optional) Git and GitHub Setup and Practice

In many of the Engineering courses, you will use git and GitHub to manage your source code. Therefore, it will be helpful to get familiar with these tools. For those of you who have never used these tools before, git is a command-line tool for synchronizing source code “repositories” on different machines. GitHub, on the other hand, is a cloud service that hosts git repositories and provides some additional tools for collaboration on code, centered around git repositories. This is one of the most popular combinations of tools for managing and collaborating on source code in the software development world, so learning to use them early in your software development career will be a great benefit.

Step 1: Download and install git on your laptop

Note: **If you're working on one of the ENGR flip servers, you can skip this step**, since those machines should already have git installed.

Git is a tool you run on your computer, but you'll need to make sure you have it downloaded and installed to be able to run it. You can download the latest version of git for any operating system here: <https://git-scm.com>. If you're on MacOS, you can also install git via Homebrew, if you use it. If you're on Linux, you can install git using your distribution's package manager (e.g. apt or yum).

Step 2: Open your command line terminal

Git is a command line program, so open up your favorite command line program, i.e., log onto flip server using MobaXterm, or:


Windows:

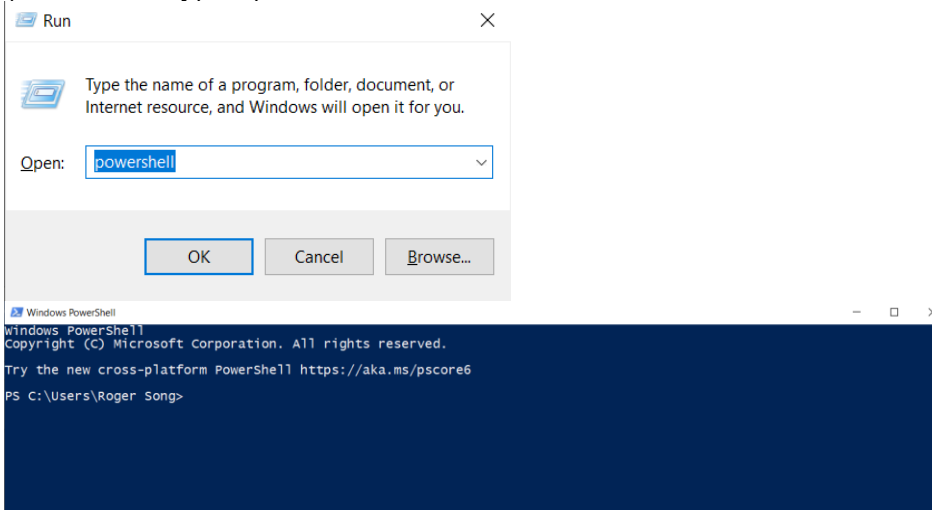
If you're using Windows, you have a few options for the command line.

If you installed git from git-scm.com, you should have a program called Git Shell installed, which you can access from your Start menu. You can run this to get to a command line.

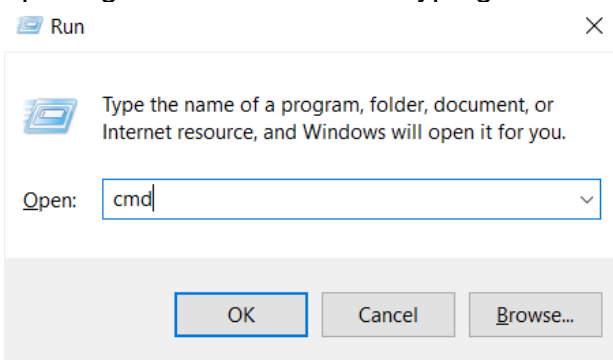


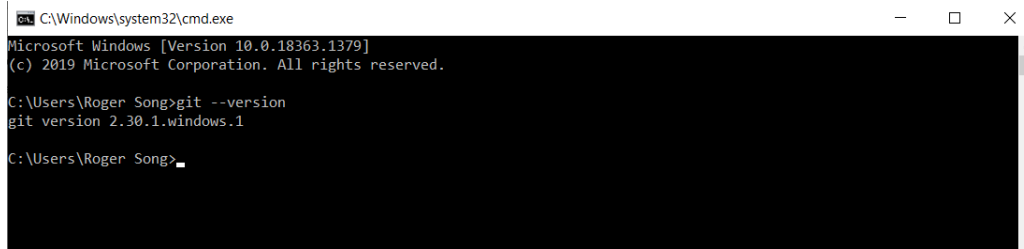
```
MINGW64/c:/Users/Roger Song
Roger_Song@Roger-Laptop MINGW64 ~
$
Roger_Song@Roger-Laptop MINGW64 ~
$
Roger_Song@Roger-Laptop MINGW64 ~
$ git --version
git version 2.30.1.windows.1
Roger_Song@Roger-Laptop MINGW64 ~
$ |
```

You can also use Power Shell, which should come already installed on Windows. ( (windows key) + r)



You can also use the standard Windows command prompt, which you can access by opening the Start menu and typing cmd in the Search/Run box.





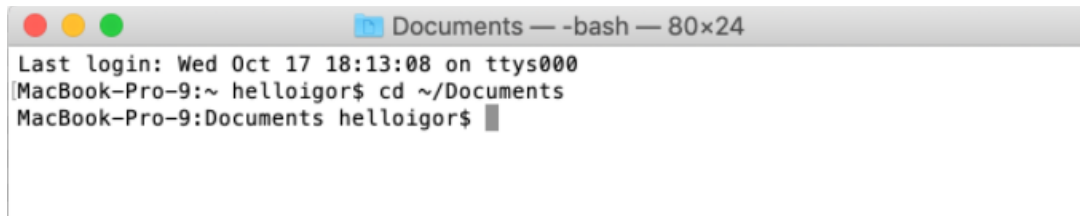
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Roger Song>git --version
git version 2.30.1.windows.1

C:\Users\Roger Song>
```

Unix (MacOS/Linux)

If you're using a Unix operating system (MacOS or Linux), you can use the terminal program that comes installed with your OS. In MacOS, this is called Terminal. In Linux, this can be called Terminal or Console.



```
Documents — -bash — 80x24
Last login: Wed Oct 17 18:13:08 on ttys000
MacBook-Pro-9:~ helloigor$ cd ~/Documents
MacBook-Pro-9:Documents helloigor$
```

Step 3: Make sure git is installed and set some basic configs

Now that you're at your command line, let's make sure git is properly installed. Run the following command:

```
git --version
```

You should see your git version printed out.

Now that you know git is installed, let's sign up your GitHub account at <https://github.com>. (You may skip this if you already have a GitHub account).

Once you have your account, return to your command line terminal and let's set some basic but necessary git configurations. Run these two commands to make sure git knows your GitHub username and email address, so it can record this information when you commit code:

```
git config --global user.name "Your Name"
git config --global user.email "your-email@somewhere.com"
```

You should also run this command to make sure git handles line endings correctly on your OS:

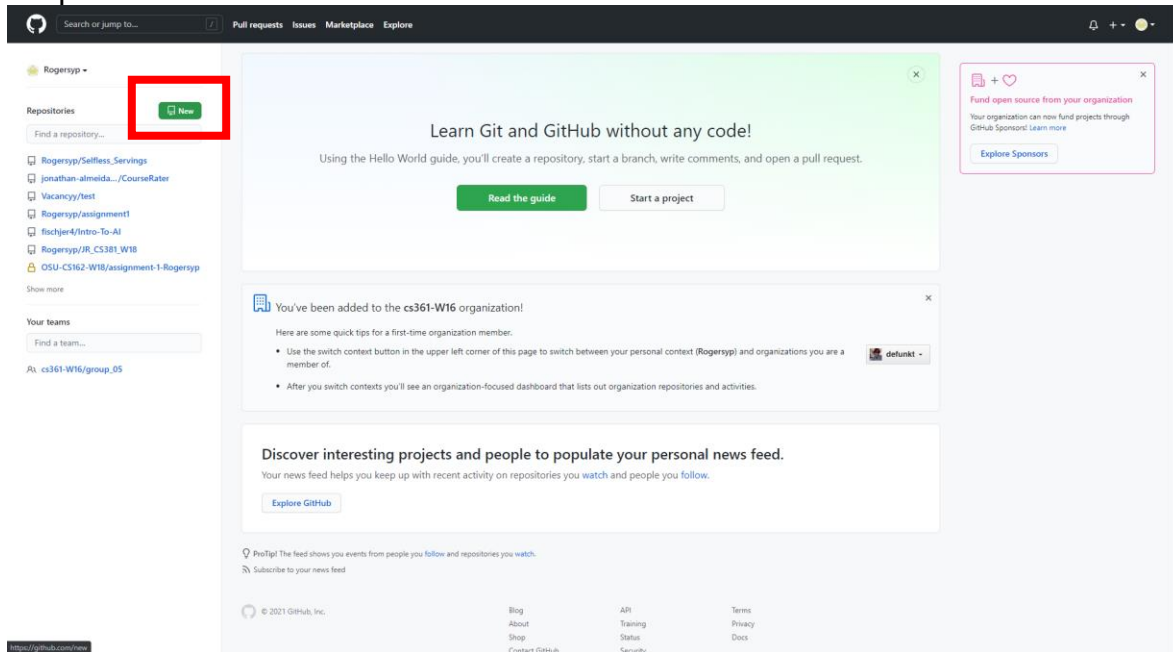
```
(Windows)          git config --global core.autocrlf true
(MacOS/Linux)     git config --global core.autocrlf input
```

You may always check your git configurations using the following command:

```
git config --list
```

Step 4: Create a repository and download your code

Let's create a repository using GitHub. On your homepage, click "New" next to "Repositories"



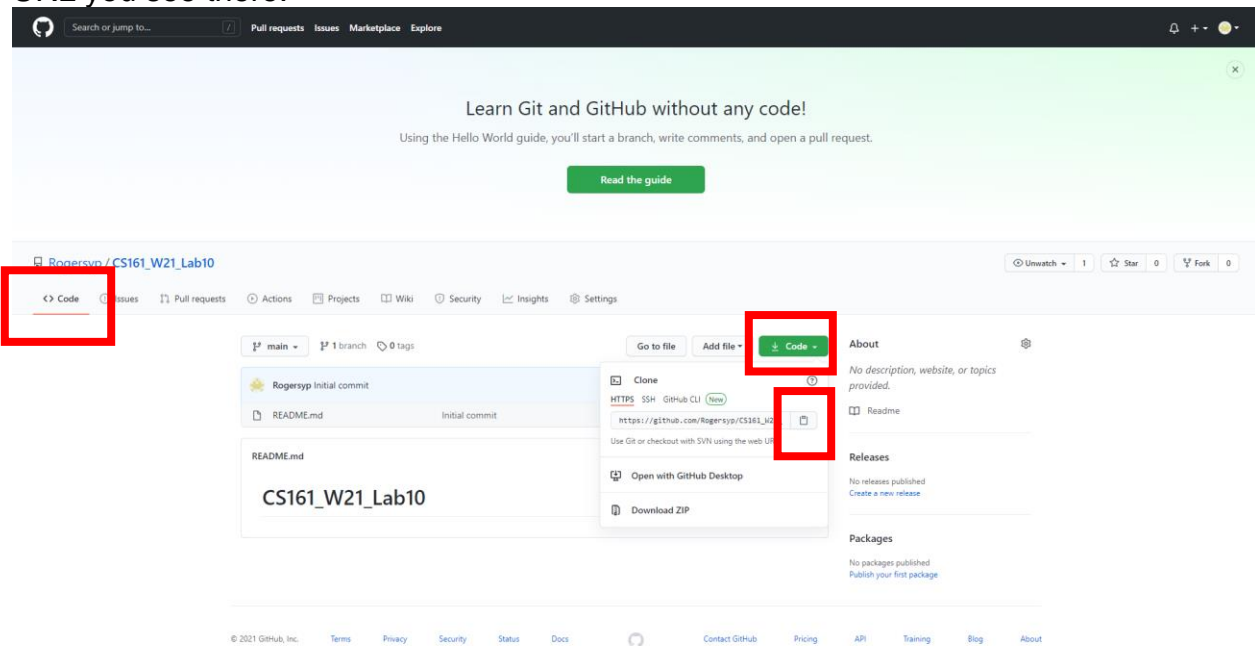
Then, name your repository, i.e., CS162_Sp23_Lab10. (It would be nice to add a README file, so you may add a description for your project). Then click "Create Repository".

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

A screenshot of the GitHub 'Create a new repository' form. The 'Owner' is 'Rogersyp'. The 'Repository name' field contains 'CS161_W21_Lab10' and is highlighted with a red box. Below it, a message says 'CS161_W21_Lab10 is available.' The 'Description (optional)' field is empty. The 'Public' radio button is selected. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked and highlighted with a red box. At the bottom, the 'Create repository' button is highlighted with a red box.

Once the repository is successfully created, it should navigate you to the “Code” page. Under “Code” button (older version should say “clone or download”), copy the clone URL you see there.



Now, go back to your command line, and cd to the directory where you want to make a copy of your repository. Once you’re there, run this command, replacing REPO_URL with the clone URL you copied just a minute ago:
`git clone REPO_URL`

```
flip3 ~/cs161/w21 164% ls
assignment1 assignment2 assignment3 cs161_w21 demo
flip3 ~/cs161/w21 165% git clone https://github.com/Rogersyp/CS161_W21_Lab10.git
Cloning into 'CS161_W21_Lab10'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
flip3 ~/cs161/w21 166% ls
assignment1 assignment2 assignment3 cs161_w21 CS161_W21_Lab10 demo
flip3 ~/cs161/w21 167%
```

This will make a local copy of your remote repository (the one on GitHub). This local repository is where you will do your work editing, adding, and deleting files. You’ll see a little later how to “push” the changes you make in this local repository back into your remote repository on GitHub.

Note: if you have trouble cloning your repository, try using the URL you get after clicking the “Use HTTPS” link under the “Code” button on GitHub. If you do this, you’ll need to make sure you enter your GitHub credentials when prompted.

Once you have your repository cloned locally, cd into the repository directory that was just downloaded and run this command:

```
git status
```

This is a git command you can use to check the current state of your local repository. It will tell you things like the name of the branch you're working in (no need to worry about this for now) and what files have changed since you last committed files into your repository (more on this later). We'll see `git status` again in a bit.

```
flip3 ~/cs161/w21 167% cd CS161_W21_Lab10/  
flip3 ~/cs161/w21/CS161_W21_Lab10 168% git status  
# On branch main  
nothing to commit, working directory clean  
flip3 ~/cs161/w21/CS161_W21_Lab10 169% █
```

Step 5: Develop source code, i.e., programming

Usually this is the step where you develop your source code inside the git repo, but for this lab's purpose, let's copy the code you wrote for the interview questions above and paste into the repository using the following command (replacing `FILE_PATH` with your path to the lab10 folder):

```
cp -r FILE_PATH/ ./
```

```
flip3 ~/cs161/w21/CS161_W21_Lab10 186% cp -r ~/lab10/ ./  
flip3 ~/cs161/w21/CS161_W21_Lab10 187% ls  
lab10  README.md  
flip3 ~/cs161/w21/CS161_W21_Lab10 188% █
```

Step 6: Commit your program into your local git repository

When you have your program working the way you want to, it's a good time to commit it into your local git repository. First, run `git status` again to see what state your repository is in. You should see output something like this:

```
flip3 ~/cs161/w21/CS161_W21_Lab10 188% git status  
# On branch main  
# Untracked files:  
#   (use "git add <file>..." to include in what will be committed)  
#  
#       lab10/  
nothing added to commit but untracked files present (use "git add" to track)  
flip3 ~/cs161/w21/CS161_W21_Lab10 189% █
```

This is telling you git sees that you created a new file/directory named `lab10`, but that file is not yet being tracked by git. We want git to track this new file. To achieve this, run this command:

```
git add --all
```

The above command is to add all files in this working repository, you may also do `git add FILENAME` if you only want to add one file.

Now, if you run `git status` again, you should see output something like this:

```
flip3 ~/cs161/w21/CS161_W21_Lab10 195% git status
# On branch main
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   lab10/makefile
#       new file:   lab10/memory.cpp
#       new file:   lab10/memory.o
#       new file:   lab10/solution.cpp
#       new file:   lab10/solution.o
#       new file:   lab10/task1
#
```

This is telling you these new files are staged to be part of your next commit. In other words, the next time you tell git to make a commit, these new files will then be tracked by git. Let's do that:

```
git commit -m "Add new files for lab10."
```

Here, we're making a commit (that is, taking a snapshot of the changes we made to our code and adding that snapshot permanently into our local git database), and we're using the `-m` option to add a message describing that commit. If you omitted the `-m` option and the subsequent message argument, git would open a text editor for you to type in a commit message.

You can verify that your commit worked by running the command `git log`, which lists all of the commits in the history of your repository. At the top of the output, you should see a commit with the message you just entered above.

Step 7: Push your program to your remote repository on GitHub

If you go back to your browser and refresh the page containing the "Code" tab in your GitHub repository, you'll see that it still doesn't contain the new file you created. This is because git doesn't perform any kind of network operations unless you explicitly tell it to. The `git commit` command is not a network operation: it simply updates your local repository.

To update the remote repository on GitHub, you need to "push" the changes from your local repository into that remote repository. Luckily, because of the way we used `git clone` to create our local repository, pushing changes back to the remote repository on GitHub is easy. All you need to do is run this command:

```
git push
```

After that command finishes, you should be able to refresh the page containing your GitHub repository and see your new file there. This is a great thing! It means your file is safely stored on GitHub.

You may view all commits by clicking on "Insights" tab, and then click "Network":

[Read the guide](#)

Rogersyp / CS161_W21_Lab10




Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security **Insights** Settings

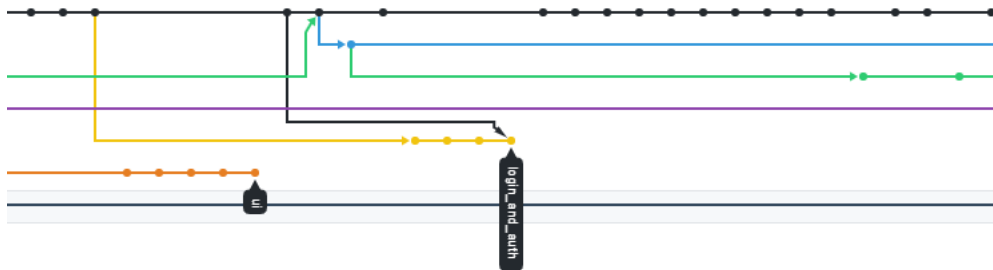
- Pulse
- Contributors
- Community
- Traffic
- Commits
- Code frequency
- Dependency graph
- Network**
- Forks

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

Owners	Mar
Rogersyp	<div style="border: 1px solid red; padding: 2px;">  8 1a855f5  </div>
	<div style="border: 1px solid red; padding: 2px;">  Rogersyp Add new files to lab10 </div>

In future CS courses, when you have collaborators working on the same project, this graph will look something like this (but don't worry about this now):



The workflow here is important to remember. This is the essential git/GitHub workflow:

1. Create a local git repo (in our case by copying a remote one on GitHub using `git clone`).
2. Modify (add, edit, or delete) files in your local repo.
3. Use `git add` to stage a snapshot of your changes to be committed to your local repo.
4. Use `git commit` to commit the snapshot of your changes to your local repository, making it permanent there.
5. Use `git push` to push your changes to your remote repository on GitHub.
6. Repeat from 2.