CSCI 315: Data Structures

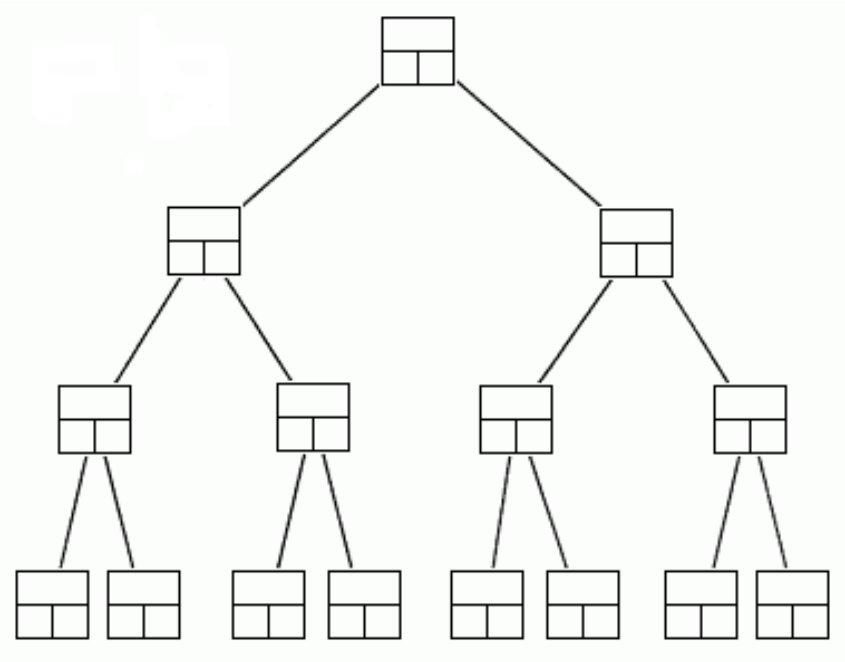
Paul E. West, PhD

Department of Computer Science
Charleston Southern University

January 7, 2019

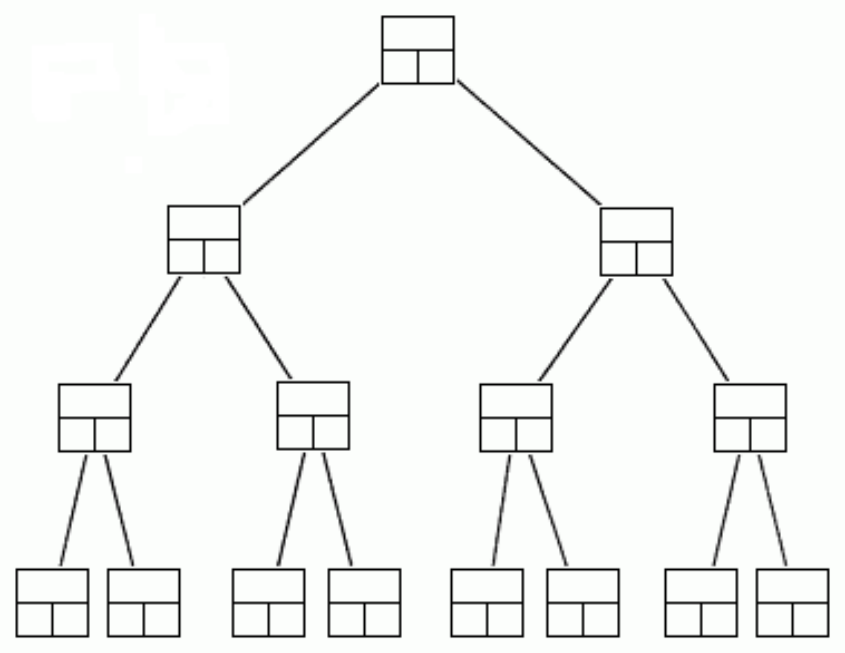
What is a Data Structure?

- A data structure is a scheme for organizing data in the memory of a computer.
- Some of the more commonly used data structures include lists, arrays, stacks, queues, heaps, trees, and graphs.



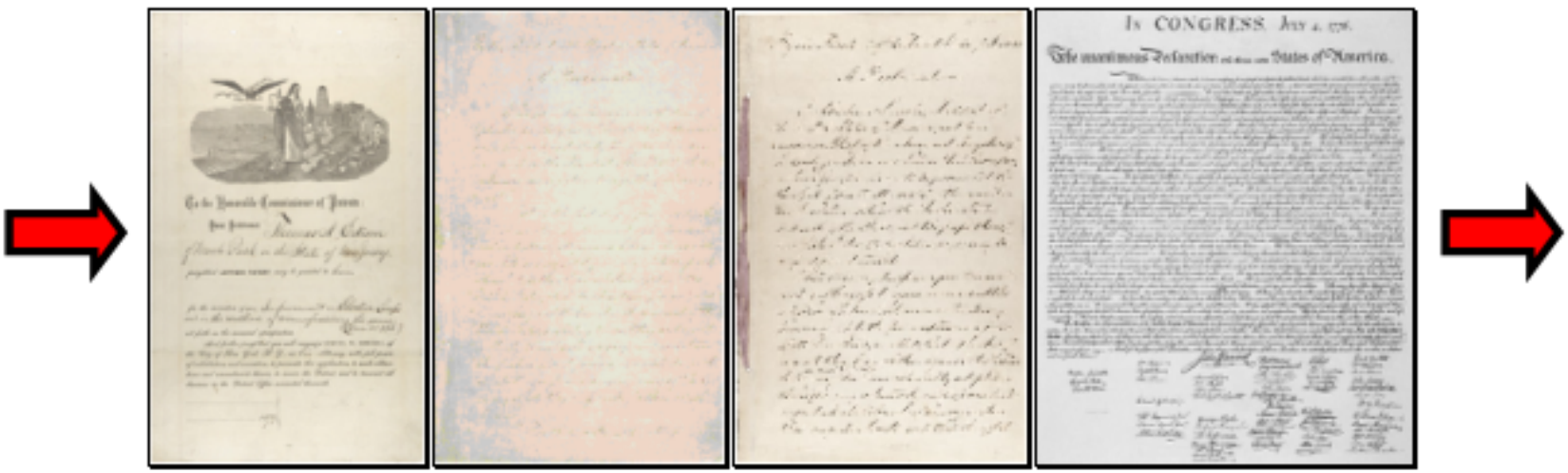
Why?

- The way in which the data is organized affects the performance of a program for different tasks.
- Computer programmers decide which data structures to use based on the nature of the data and the processes that need to be performed on that data.



Queue

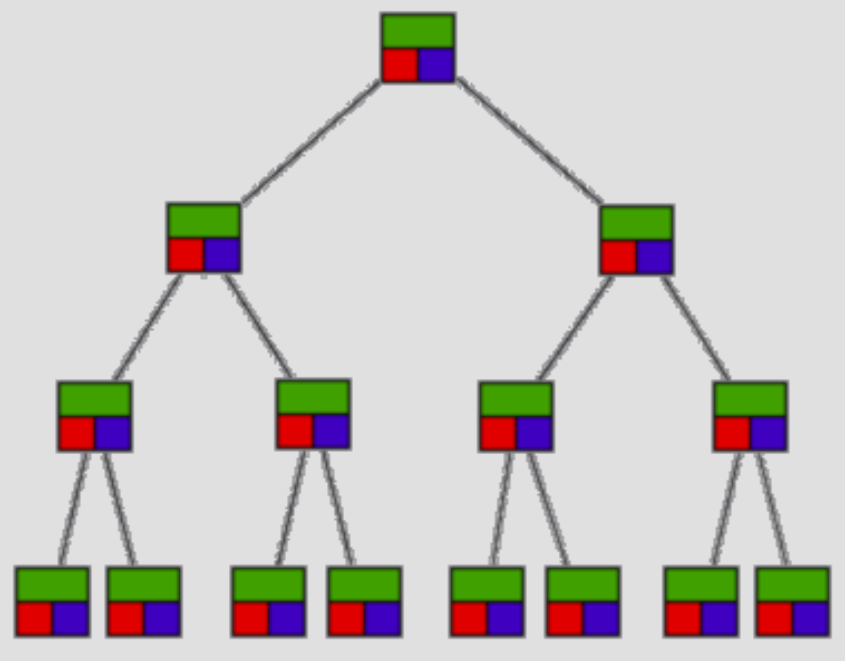
- A queue is an example of commonly used simple data structure. A queue has beginning and end, called the front and back of the queue.



- Data enters the queue at one end and leaves at the other. Because of this, data exits the queue in the same order in which it enters the queue, like people in a checkout line at a supermarket.

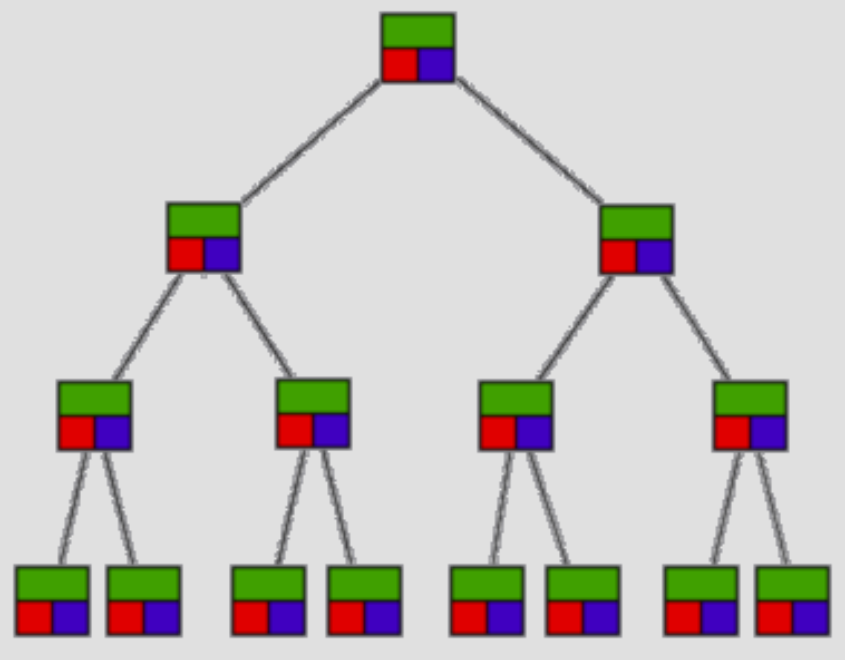
Binary Trees

- A binary tree is another commonly used data structure. It is organized like an upside down tree.
- Each spot on the tree, called a node, holds an item of data along with a left pointer and a right pointer.



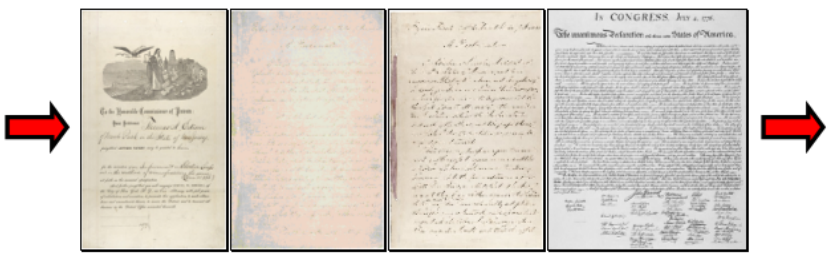
Binary Trees

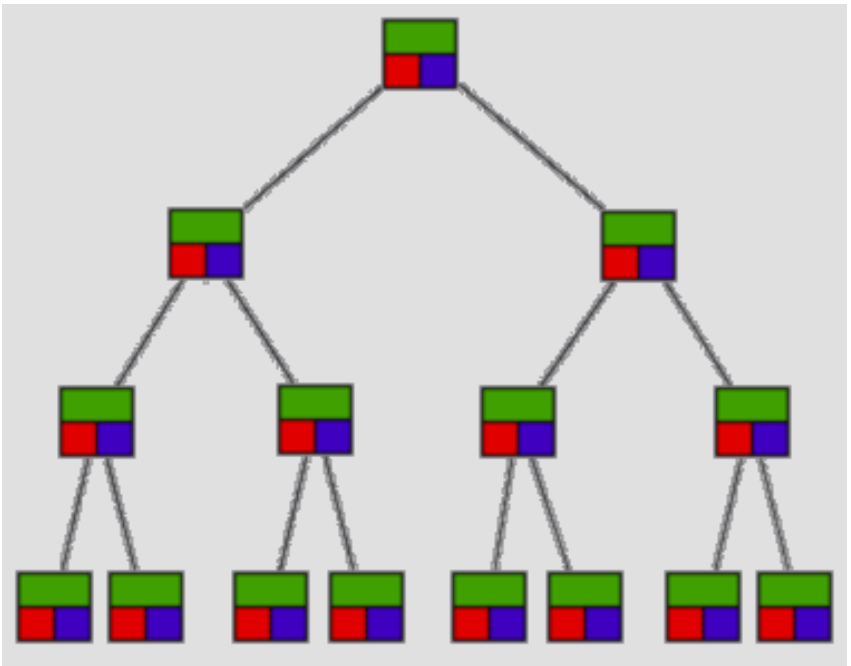
- The pointers are lined up so that the structure forms the upside down tree, with a single node at the top, called the root node, and branches increasing on the left and right as you go down the tree.



Decisions Decisions...

- When do we choose one over the other?





About the Professor

- PhD from Florida State University in Computer Science
- Faculty Experience:
  - Charleston Southern: Assistant 2015-Present
  - College of Charleston: Adjunct 2013-2014
- Work Experience:
  - Naval Research Lab: (2018 - Present)
  - Google (2014): Android Bluetooth/Wi-Fi/Telephony
  - SPAWAR/Government Contracts (2009-2014, 2017): Communication systems
  - DenimGroup (2004-2005): Start-up; web design and network security

First Goal:

- Create/Finish the solid foundation for Computer Science
  - Parable of Wise/Foolish Builder: Matthew 7:25-27.
  - Labs before DS: Copy/Pasta class code, rework it, and submit.
  - Now: You have a bag of tricks (tools/data structures), solve this problem!
- How I assess:
  1. Solve numerous programming problems
  2. Fix broken code
  3. Fix performance of existing code
  4. Show me you understand how a computer assembles your program

Other Goals

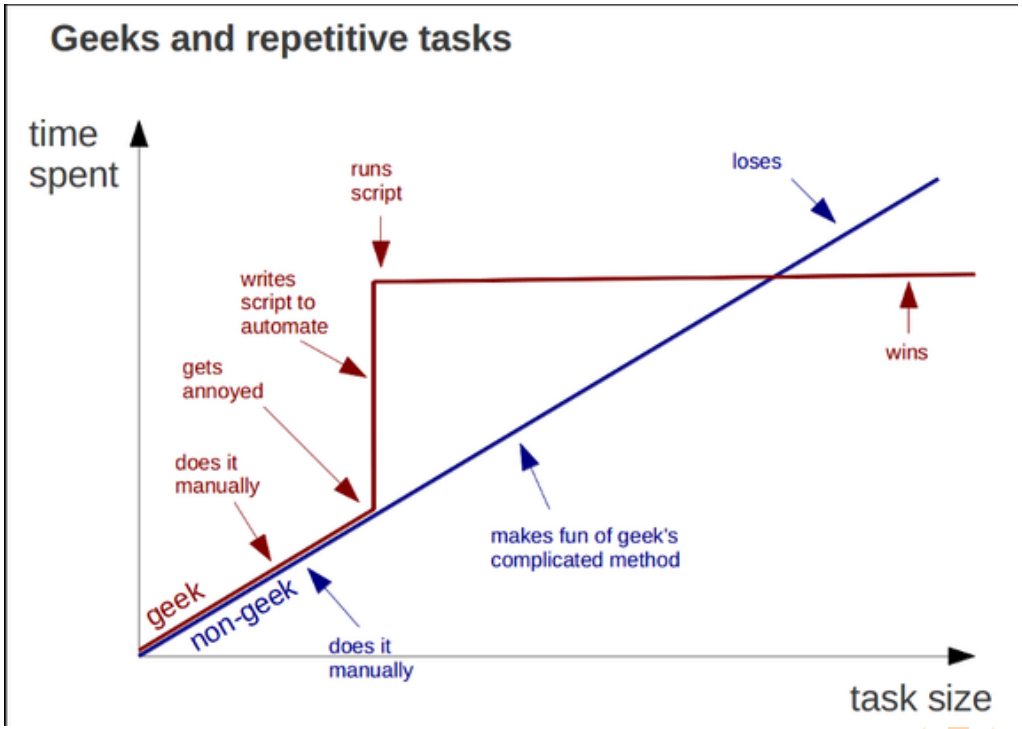
- Analyze a problem: find errors, and performance issues.
- Generate reports explaining performance issues and solutions.
- Demonstrate a basic understanding of *nix environment.

*nix Environment

- We are using a *nix Environment. I am using Debian.
- You are free to use any *nix environment, but a Linux derivative is **highly** recommended.
- I recommend installing VirtualBox and run Ubuntu.
- Make sure you **INSTALL** Ubuntu.
- Another option is using Bash on Ubuntu on Windows 10 (with creator's update).
  - Versions on or after 1703 can install Ubuntu 16.04. Make sure your Ubuntu version is **16.04** or later!
  - Please Google how to install, there are many ways.
  - You may use the machines here in class with Bash for Windows, but you must install bash for you user.
- Other options:
  - Mint, Ubuntu, Fedora, Centos, etc...: work fine
  - OSX: I have seen it work, with much difficulty
  - Windows CLI: Only sadness has occurred.
  - BSD & Solaris: I haven't tried, but it should be fine

CLI Environment

- We will also be using the Command Line Interface (CLI).
- Why?
  - It forces you to understand more of what is actually happening.
    - Most GUIs are “paint” on top other programs
  - For many tasks, you can accomplish a lot more is less time.



Passing Data Structures (DS)

- DS is a *marathon*, **not** a *sprint*!
- The numerous labs break up the work.
- Last semester was 24 labs and 4 projects.
- **Warning:** *The pass rate of a student becoming 3 weeks behind is 0%*.
- If you fall behind, be sure to see me!
- Advice from a student that took DS, Calc II, and football in the same semester (and passed everything):
  - Plan to do a some DS **and** Calc II each day
  - Do not get behind
  - If you cannot get lab done, make sure your lab compiles and submit to get what points you can.
  - If you fall behind, don't try to catch up, take the 0 and move on.

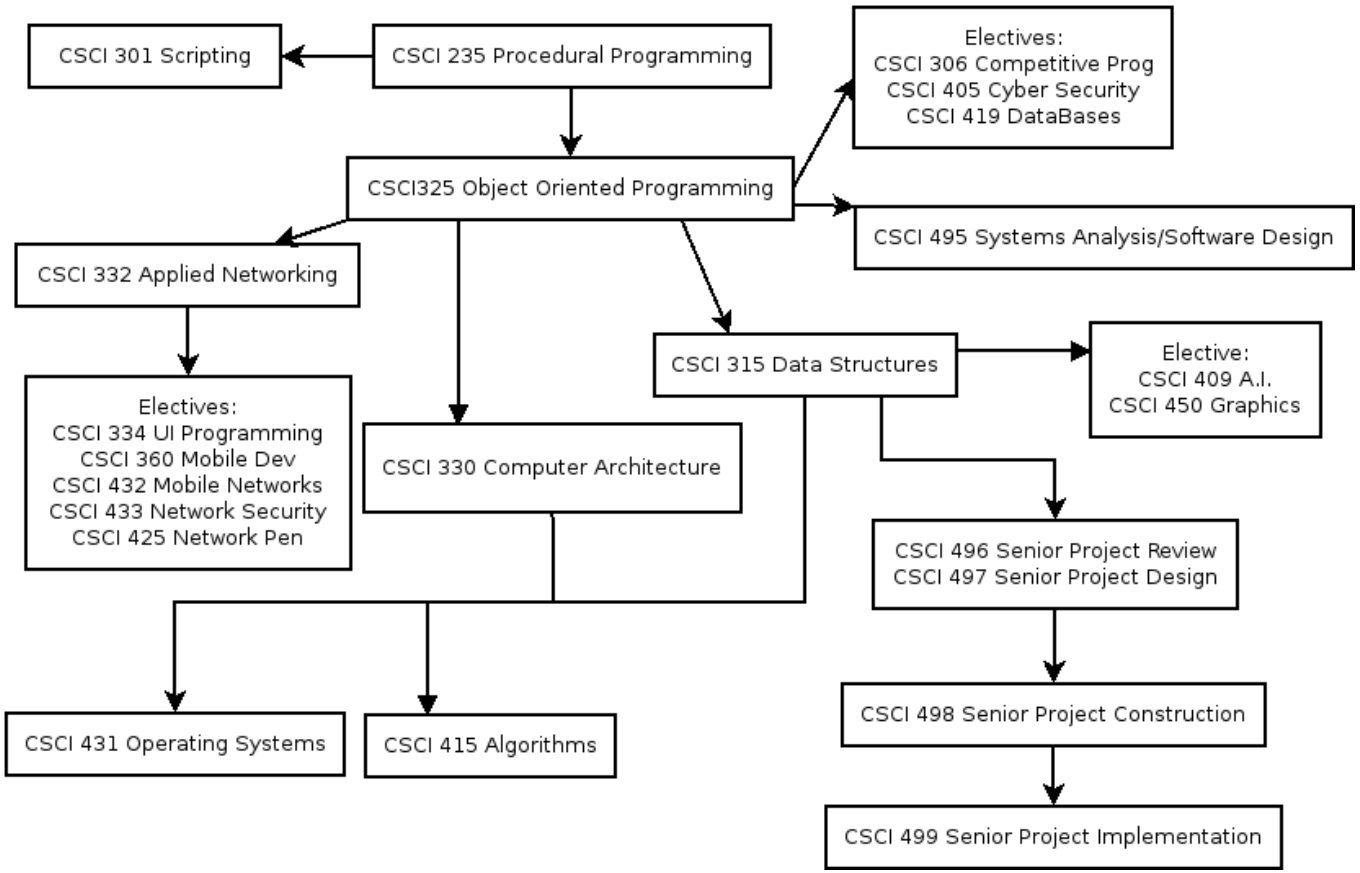
Syllabus

Lets go over the syllabus...

Warning:

- Most students consider this class to have the highest workload.
- This class really prepares your for work experience.

Do Not Drop!


CSCI 301 Scripting
CSCI 235 Procedural Programming
Electives:
CSCI 306 Competitive Prog
CSCI 405 Cyber Security
CSCI 419 DataBases
CSCI325 Object Oriented Programming
CSCI 495 Systems Analysis/Software Design
CSCI 332 Applied Networking
CSCI 315 Data Structures
Elective:
CSCI 409 A.I.
CSCI 450 Graphics
Electives:
CSCI 334 UI Programming
CSCI 360 Mobile Dev
CSCI 432 Mobile Networks
CSCI 433 Network Security
CSCI 425 Network Pen
CSCI 330 Computer Architecture
CSCI 496 Senior Project Review
CSCI 497 Senior Project Design
CSCI 431 Operating Systems
CSCI 415 Algorithms
CSCI 498 Senior Project Construction
CSCI 499 Senior Project Implementation

Expectations

CSCI 215/217	Basic computer use and the ability to modify existing simple programs
CSCI 235	Can create simple programs from well defined specifications and solid test cases
CSCI 325	CSCI 235 plus the ability to work in groups.
CSCI 315	Can create both simple and somewhat complex programs from a well defined specification, and creates own test cases.
CSCI 431	CSCI 315 plus able to derive specification from vague requirements and perform basic research of emerging technologies.

Bash for Windows

- Our lab has Windows 10 with Creator's Update (1703) and Linux subsystem enabled.
- You will need to install Bash For Windows.
- Please login with your standard CSCI login.
- I will go through how to install Bash for Windows.

Terminal

- terminal: (terminal emulator), is a text-only window in a graphical user interface (GUI) that emulates a console.
- It is were we use our CLI
- xterm, gterminal, lxdeterminal
- Run whatever you want!

- Developed by Linus Torvalds
- Version Control Tool
  - Subversion
  - CVS
- Open Source
- Free!
- Just a folder in a directory

Git Golden 5

- clone/init
- Add
- Commit
- Push
- Pull

Joining the Class

- Create a GitHub account (or use an existing one) and login.
- email me you username: pwest@csuniv.edu
- You should (eventually) receive an email to join CSU
- Click the link and join
- Click on the csci-315-spring-2019 repository
- Click fork
- Now clone your repository:
  $ git clone <your repo>
- Lets look through the repository...

Auto Grader

- I will attempt to maintain an auto grader for this class.
- It will function similarly to the one used in CSCI 325 except:
  - It will not say what test case(s) you failed
  - It will not mention if there is a timeout
  - It will not say if there are compilation error(s)
  - It *may* tell you which part of the lab you got right or wrong
- **Note:** I will **NOT** hand out the test cases from the auto grader. It is imperative that you learn how to write your own test cases!
- The grade from the auto grader should be pushed to your repository after submission.