About this Workbook

> This workbook is designed for the Tapia 2021 Workshop facilitated by Sarah M Brown and Victoria Chávez. If you did not attend the workshop, this workbook can still provide resources and guidance on equitable assessment!

If you prefer, download a pdf

Workshop Overview

In this workshop, we will discuss assessment from the assignment level to the course level that aligns with learning and equity. We will first outline key terms, show how assessment can encourage learning and support equity. Next, participants will practice applying the new terminology and theoretical grounding by studying examples. Finally, participants will get to revise materials for their own courses.

Using the Workbook

If you're familiar with Github, you can fork this workbook's repo to work through the case studies and templates. You can also copy-paste the prompts and examples onto your editor of choice or even print the pages to work offline (**workshop attendees**: please don't work offline during the workshop).

For Workshop Attendees

During the Workshop, we will guide you through several activities (see our full agenda) to modify your course(s) to better motivate and engage students via centering learning and equity in your courses' assessment. **Please keep your work in an virtually-shareable format for the peer feedback portion of the workshop.**

For Workshop Non-Attendees

If you weren't able to attend the workshop, you can use our detailed activity information, case studies, and templates to revise your syllabus and/or assignments.

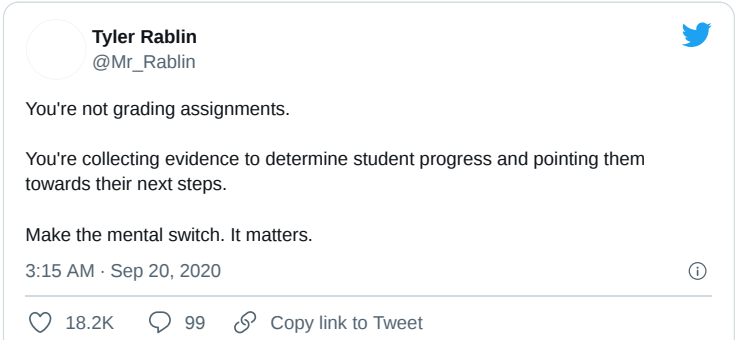
Contributing to the Workbook

Contributions are welcome! To submit revisions, additional case studies or templates, or resources, please submit a pull request on Github.

Introducing alternative forms of assessment

Assessment can have a big impact on our students. While students often view the work they do in class as a means to a grade, our goal as instructors is typically that the activities we ask them to do help them learn.

We can focus the way we evaluate that work to center students learning to help direct their attention to their own learning.



Tweet your reply

Using Case Studies

The **Case Studies** section contains examples of assignments and (the grading section of) syllabi that demonstrate some principles for grading that support learning and equity.

> Workshop Attendees: In your breakout group, take about 10 minutes to discuss **one assignment and one grading scheme** from the **Case Studies section**. Write any questions that come up and we'll address them in the workshop and/or add them to our FAQs.

For each case study, answer the following questions:

1. Which principle(s) are applied?
2. How is the principle you identified applied?
3. What do you like about this case?
4. What would you want to change?
5. What questions do you have about case study?

Using Templates

This section of the workbook is designed to support you in applying the principles discussed previously to your own course(s).

> Workshop Attendees: Take about 10 minutes to complete the steps below. **Please keep your work in an virtually-shareable format.** In the next part of the workshop, you will receive and provide peer feedback.

1. Decide if you will revise assignments but keep your grading scheme **or** will focus on revising your grading scheme and policies and have assignments follow.
    - Keep in mind that revising your grading scheme will most likely mean eventually having to modify each assignment's grade breakdown for it to make sense with the overall scheme.
2. Choose any of the templates or guided design documents under the **Templates** section.
    - Make sure to keep notes for yourself as well as a before and after copy of your syllabus and/or assignment(s).
3. Consider 1-2 questions or concerns you have to focus your peer feedback so that it's the most helpful.

Peer Feedback

Peer feedback is crucial in course development and refinement. Your peers may notice or think about things you haven't considered and they may have resources or advice to share!

> Workshop Attendees: In your breakout group, take about 10 minutes to take turns describing the changes you've started working on, the questions or concerns you have, and providng feedback for your peers. If you're able to share your work via a link, file, or screenshare, please do so. Provide any feedback or advice you think will be most useful for your peers. Make sure to keep it concise so everyone is able to share!

Reflection

After receiving feedback, it's good practice to reflect on what we've changed and what we've learned before piling on more changes. As questions come up, we should also do our due diligence to have those questions answered by our peers, others who have done similar work, or teaching adn learning experts at our institution's center for teaching and learning.

> Workshop Attendees: As a whole group, we will take about 15 minutes to reflect on your modifications, the feedback received, and the questions you have for moving forward. Please summarize your reflections as best as possible into: 1 highlight, 1 lowlight, 1 question or concern. Presenters will pose reflective exercises and answer participants' questions live and/or add them to our FAQs.

Reflective Questions to Consider:

1. What are your overall thoughts on starting these revisions?
2. How was the peer review? Did you learn something helpful?
3. What is the one most important change you plan to make to your course based on the workshop today?
4. What questions or concerns do you have?
5. Are there any 'blocks' preventing you from making updates to your course(s)?

Scheme: Competency-Based Grading in Programming for Data Science

Full Sites:

- Fall 2020
- Fall 2021

This course uses a competency-based grading scheme and assesses assignments on specification agains an overall course rubric.

Instructor Process

I broke down the course learning outcomes into 15 component skills and wrote three outcomes for each describing a skill acquisition expected for each skill. This describes each of the 45 (15*3) achievements that constitute the students' grades.

System basics

To earn an A required all 45 achievements, a B required all level 1 and level 2 achievements, a C required all level 1 achievements. Students could earn level 1 achievements in any activity: class participation, weekly programming assignments, or a portfolio submitted 4 times through the semester.
Level 2 achievements could be earned only on assignments and portfolio checks.
Level 3 achievements could only be earned on portfolio checks.

In class questions checked basic understanding through multiple choice questions and short programming problems. These were cumulative only, not graded as a percentage correct. Assignments were designed to assess level 2 achievements, such that each skill was assessed in at least two assignments. Portfolios were open ended, encouraging students to use general prompts to explore the material deeper than was covered in class, guided by the level 3 achievement definitions.

Equity and learning focus

- the grade was not based on averaging performance across activities; the basis was the material only.
- Students had at least 2 chances (and mostly many more) to earn every single achievement. A student could require 2 attempts at every single achievement and still earn an A.
- Students could skip assignments as they deemed necessary for their regular schedule.
- grading was that they had to demonstrate understanding: both applying material and communicating well enough.
- Because the grade was cumulative, students focused on their learning trusting that getting things eventually would be worth it

Assignment: Data Science Portfolio

live site

> **Note**
>
> Students submitted their portfolio as a jupyter book via GitHub classroom. They added more content to the same repository over the course of the semester.

Prompts

Overall

The prompts provide a starting point, but remember that to earn achievements, you'll be evaluated by the rubric. You can see the full rubric for all portfolios in the syllabus. Your portfolio is also an opportunity to be creative, explore things, and answer your own questions that we haven't answered in class to dig deeper on the topics we're covering. Use the feedback you get on assignments to inspire your portfolio.

Each submission should include an introduction and a number of 'chapters'. The grade will be based on both that you demonstrate skills through your chapters that are inspired by the prompts and that your summary demonstrates that you know you learned the skills. See the formatting tips for advice on how to structure files.

Correct an Assignment

Choose an assignment that you did not achieve the target level for. Write a blog style notebook analysis that corrects what you could have done better, what you learned, and addresses the misconception if applicable.

> **Note**
>
> To get credit for this, student had to not only correct, but truly reflect and explain.

Deeper Analysis

For one of the assignments, if there was something you were curious about. Try it out and investigate how to answer it. Vary parameters and document your investigation.

New Analysis

For a topic of interest, clean, explore and model the data. Work with messy data or data provided in multiple files to earn prepare and construct achievements or use clean data to earn only EDA and modeling achievements.

CheatSheet

Make a cheatsheet with examples of the several different parameter settings for common operations for one topic.

This cheatsheet is an example, it's too broad, but it's the same idea. Yours should cover one topic in greater detail and demonstrate your skill according to the rubric.

Scheme: Competency-Based Grading in Data Structures

This course example uses a competency-based grading scheme and assesses assignments on specification against an assignment rubric. Please visit the Spring 2021 course site for more detailed information.

Instructor Process

The course covers a very large number of topics over a 13 week period, several of which previous instructors didn't get to as students fell behind throughout the semester. I decided to pick the most important topics and classified them as non-negotiables– these were topics a student shouldn't leave a data structures and algo course without understanding. I chose to have 4 levels of knowledge, of which the highest required a self-directed final project to demonstrate. I categorized the original assignment requirements into the three levels of knowledge left, and that became that assignment's rubric.

A concrete example using writing: Say the learning objective of the assignment is to write a paragraph. To reach awareness, a student would have to write between 3-5 sentences total. To reach this level of knowledge (awareness) students don't have to worry too much about the sentences flowing well and being related, they just have to meet the bare minimum of length. To demonstrate understanding, the 3-5 sentences must be on the same topic, and lastly, to demonstrate competency, students would have to write 3-5 sentences about the same topic, that flow well together, and form a unified single idea.

System basics

Almost each grade band (A, B, and C) had two options/avenues to be earned (D only had one path). One of the options offered more flexibility for missing or incomplete assingments, the other did not. Final projects were an optional way to demonstrate knowledge to give students who missed any assignments an additional opportunity to boost their grade. Completing and submitting a final project was also required to earn an A in the course. There was also an opportunity to gain a + designation described in the syllabus (i.e., to get a B+ instead of a B).

Equity and learning focus

- Final course grades were not based on averaging performance across assignments meaning students could have bad weeks that wouldn't hurt their final grade.
- Students had multiple opportunities to demonstrate their knowledge in any given topic area.

Other applied strategies for equity

- Assignment resubmissions were allowed and encouraged throughout the semester. Some topics take longer than others to click, sometimes students have only a limited amount of time to devote to an assignment but can return to it later.
- Soft deadlines and hard deadlines allowed students to self-pace while having an outline of where they should be at with their assignment completion.
- Open-ended final projects to allow students to apply course topics to their own interests.

Assignment: Heaps using Comptency-Based Grading

NOTE: This is an abridged version of the lab handout, to see the full version, please visit the course site.

Motivation (Why are we doing this?)

The goal of this lab is to provide background on `Heaps`.

Background Info

Please see full lab handout for this section.

Your Task

NOTE: Please see full lab handout for file downloads.

Similar to last week's lab, we give you a makefile, a unit test file, a doctest header file and a header file with class definitions for `HeapNode` and `HeapTree` classes (plus a couple other things). We have supplied `heap.cpp` to house your class definitions, and `test.cpp` will serve as your main file.

- heap.h
- heap.cpp
- test.cpp
- doctest.h
- makefile

Requirements

Your goal for this lab is to complete the following tasks **in order**:

1. Diagram the following operations for a min heap:
    - insert: 5, 3, 7, 2 , 4, 6, 8
    - remove_min
    - delete: 4
2. Implement 'min_heapify()', 'find_last()', 'remove_min()', and 'delete_element()'
3. Copy your entire solution to a new Directory, and modify everything to create a max_heap
    - You will need to re-do the test cases in `test.cpp`!

Handing in

Please call a TA over to get checked off before leaving your lab section (regardless of how far you got). If you want to continue working on your lab after your lab section, come to hours to get checked off.

Grade Breakdown

This assignment covers `heaps` and `balanced trees` and your level of knowledge on them will be assessed as follows:

- To demonstrate an `awareness` of these topics, you must:
    - Successfully meet requirements **1**
- To demonstrate an `understanding` of these topics, you must:
    - Successfully meet requirements **1 and 2**
- To demonstrate `competence` of these topics, you must:
    - Successfully meet requirements **1 through 3**

Scheme: Specifications Grading in Data Structures

This course example uses a specifications grading scheme and assesses assignments on complete/incomplete basis. Please visit the Summer 2021 course site for more detailed information.

Instructor Process

While students enjoyed a grading structure that centered learning and offered them flexibility, having topics that mattered more for their final grade created unnecessary confusion so I decided to change the grading scheme to specifications grading for the summer. For the summer, I kept assignment requirements as-is. Each assignment became complete/incomplete and the only way to complete any given assignment was by meeting ALL requirements. Final course grades were based on the total number of completed assignments, with a minimum of 14 being required to pass with a D (for reference there were around 21 total assignments given out).

System basics

Students had to meet all the requirements (I used an autograder so they had to pass all the autograder tests) of:

- 14-15 assignments to earn a D
- 16-17 assignments to earn a C
- 18-19 assignments to earn a B
- 20+ assignments to earn an A

For reference, there were a total of 21 graded/qualified assignments. Completing and submitting a final project was also a required part of passing the course. Final projects didn't count towards a student's final grade, though an exemplar final project could earn a student a + designation (i.e., a B+ instead of a B). There was an additional opportunity to gain a + designation described in the syllabus, as well as a way to earn a "freebie" (replace a mising or incomplete assignment).

Equity and learning focus

- Final course grades were not based on averaging performance across assignments meaning students could have bad weeks that wouldn't hurt their final grade.
- Students had the agency to choose which and how many assignments to complete.

Other applied strategies for equity

- Assignment resubmissions were allowed and encouraged throughout the semester. Some topics take longer than others to click, sometimes students have only a limited amount of time to devote to an assignment but can return to it later.
- Soft deadlines and hard deadlines allowed students to self-pace while having an outline of where they should be at with their assignment completion.
- Open-ended final projects to allow students to apply course topics to their own interests.

Assignment: Heaps using Specifications Grading

NOTE: This is an abridged version of the lab handout, to see the full version, please visit the course site.

Motivation (Why are we doing this?)

This lab will help you solidify your understanding of self-balancing trees, in particular heaps. You'll also implement heapsort as a part of this lab.

Background Info

Please see full lab handout for this section.

Your Task

NOTE: Please see full lab handout for file downloads.

Complete the implementation of a MIN heap. Pay attention to the **hints in the code (CPP AND H FILES)**.

1. Download the files below (it's easiest if you right click and then choose download or save)
   heap.h : Contains the **declaration** of the lab you'll be implementing. **Do not** alter any of the given function signatures.
   heap.cpp : Contains the **definition** of the lab you'll be implementing.
   test.cpp : EMPTY TEST FILE (with a template for you to follow). **You should add your own test cases to this file**.
   doctest.h : The library we'll be using for our own automated tester. There's a ton of code in here so don't worry too much about understanding everything it does– just know the library is extremely powerful and we'll only be scratching the surface of everything it can do. **Do not edit this file**.
2. Once you have downloaded all the files, inspect them. Ask questions if you have them.
3. Create your own makefile to compile and run your program.
4. As you hopefully noticed, a lot of these functions are familiar. Use your Lab 20 implementation to complete the following functions:
   - `height`
   - `preorder`
   - `inorder`
   - `postorder`
   - `destroy`
   - `isFull`
   - `isComplete`
   - `size`
5. Complete the implementation of `min_heapify`. Pay attention to the **comments and hints in the code**. Refer to the course material (slides, readings, and background info above) as needed.
   - This is a MIN heap so all children must be larger than their parents
6. Complete the implementation of `insert`. Pay attention to the **comments and hints in the code**. Refer to the course material (slides, readings, and background info above) as needed.
   - This is a MIN heap so all children must be larger than their parents

7. Add your own test cases for the functions you just completed. Run, compile, and test your program by running `make` in your terminal, while in the directory all these files are located in. Debug as needed.
8. Complete the implementation of `find_last`. Pay attention to the **comments and hints in the code**. Refer to the course material (slides, readings, and background info above) as needed.
9. Add your own test cases for the function you just completed. Run, compile, and test your program by running `make` in your terminal, while in the directory all these files are located in. Debug as needed.
10. Complete the implementation of `remove_min`. Pay attention to the **comments and hints in the code**. Refer to the course material (slides, readings, and background info above) as needed.
11. Add your own test cases for the function you just completed. Run, compile, and test your program by running `make` in your terminal, while in the directory all these files are located in. Debug as needed.
12. Complete the implementation of `delete_element`. Pay attention to the **comments and hints in the code**. Refer to the course material (slides, readings, and background info above) as needed.
13. Add your own test cases for the function you just completed. Run, compile, and test your program by running `make` in your terminal, while in the directory all these files are located in. Debug as needed.
14. Complete the implementation of `search`. Pay attention to the **comments and hints in the code**. Refer to the course material (slides, readings, and background info above) as needed.

- **HINT:** This will look more like the search from Lab 19 than the one from Lab 20.

1. Add your own test cases for the function you just completed. Run, compile, and test your program by running `make` in your terminal, while in the directory all these files are located in. Debug as needed.
2. Add a public **void** function, `heapsort`, to your H and CPP files. The function should take in an array of integers and an integer representing the size of the array.
3. Complete the implementation of `heapsort`.

- Build a heap from the numbers in the array passed in and then rewrite the contents of the array so they're in sorted order

1. Add your own test cases for the function you just completed. Run, compile, and test your program by running `make` in your terminal, while in the directory all these files are located in. Debug as needed.

Requirements

Your goal for this lab is to complete the following tasks **in order**:

1. All heap functions behave as expected for a min heap
2. Heapsort function behaves as expected

Handing in

To submit your solution to Gradescope, select **all of the following files** and use the *drag and drop* option:

- heap.h
- heap.cpp
- test.cpp

Grade Breakdown

You must successfully meet requirements **1-2** in order to receive credit for this assignment.

Scheme: Specifications Grading in Web Development

Syllabi:

- Web Design and Programming Syllabus (i.e., front-end web dev)
- Dynamic Web Design and Programming Syllabus (i.e., back-end and full-stack web dev)

These courses use a specifications grading scheme and assesses assignments on complete/incomplete basis.

Instructor Process

I used a backwards-design to design assignments and outlined all the requirements necessary to complete each assignment. Each assignment was evaluated as either complet or incomplete and the only way to complete any given assignment was by meeting ALL requirements. Final course grades were based on the percentage of completed assignments, with a minimum of 60% being required to pass with a D.

System basics

Students had to meet all the requirements of:

- 90% or more assignments to earn an A
- 80% or more assignments to earn a B
- 70% or more assignments to earn a C
- 60% or more assignments to earn a D

Completing and submitting a final project was also a required part of passing the course. Final projects didn't count towards a student's final grade, though an exemplar final project could earn a student a + designation (i.e., a B+ instead of a B). There was an additional opportunity to gain a + designation described in the syllabus, as well as a way to earn a "freebie" (replace a mising or incomplete assignment).

Equity and learning focus

- Final course grades were not based on averaging performance across assignments meaning students could have bad weeks that wouldn't hurt their final grade.
- Students had the agency to choose which and how many assignments to complete.

Other applied strategies for equity

- Assignment resubmissions were allowed and encouraged throughout the semester. Some topics take longer than others to click, sometimes students have only a limited amount of time to devote to an assignment but can return to it later.
- Soft deadlines and hard deadlines allowed students to self-pace while having an outline of where they should be at with their assignment completion.
- Open-ended final projects to allow students to apply course topics to their own interests.

Assignment: Image Processing and Optimization

Description

1. Remember from Unit 3 that all sites we build must be accessible
    1. Make sure your site is easy to navigate using the "Tab" key.
    2. Make sure your site is easy to understand using a screen reader.
2. Make sure you're following copyright laws with all images used on your site:
    1. Are you giving appropriate credit?
    2. Are you using royalty-free images?
    3. Are you using images you can modify? (You'll be modifying most images for this lab)
3. Use Pixlr (or photo editing software of choice) to do the following:
    1. Create a banner image for your site
    2. Create a logo for your site
    3. Choose an image on your site and remove the background. Save it using the appropriate format for it to have a transparent background.
4. Add your banner, logo, and transparent image to your site.
5. Time how long it takes for your each page to load on your site (write it down).
    1. Use "Inspect element" and click on "Network" to see how large each file is, how long it took for each element to load, and how long it took for your entire site to load.
    2. IMPORTANT: Test your loading time in a NEW browser AND in Incognito/Private Browsing mode
6. Optimize all the images on your site so each page on your site loads in no more than .3 seconds (300 milliseconds).

    1. Be careful to balance quality and size– you don't want blurry images on your site!
    2. IMPORTANT: Test your loading time in a NEW browser AND in Incognito/Private Browsing mode
7. Upload all necessary files to cPanel and make sure all the images and links work, and that your site looks the way you expect it to.
    1. Create a text post on Tumblr with the following:
    2. Your custom homepage (using your URI-given domain)
    3. Answers to the following questions:
        1. How many, if any, images did you have to replace due to not following copyright laws?
        2. What, if anything, did you have to change to make your site accessible?
        3. What tools (e.g., eraser, magic wand, crop, etc.) did you use to remove the background of the image you selected?
        4. How did you optimize your images?
        5. How long did it take your site to load pre and post optimization?
8. Define, in your own words, the following terms in your dictionary:
    1. TIFF
    2. JPEG
    3. PNG
    4. GIF
    5. RAW
9. Link your post as Lab 8 on your Labs page

Overview: What to submit

1. Link to your Lab 8 post

Requirements

1. Post is linked on Labs page
2. Post includes link to site
3. Post includes answers to all (5) questions
4. Each page on site loads in .3 seconds or less
5. Site is accessible (tab key navigation and screen reader)
6. Site includes custom-made banner, logo, and transparent image
7. All image sources are somewhere on site
8. All links work
9. Correct HTML and CSS syntax
10. No console errors or warnings
11. Portfolio has all (5) terms defined in dictionary

Syllabus: Skill Acquisition Rubric

Building out this rubric is the first step in designing a competency-based grading scheme like Sarah's Programming for Data Science

Keyword	Skill	Level 1	Level 2	Level 3
a single word	multiworkd phrase that summarizes	a learning outcome at the basic level	a mid-level learning outcome	an advanced level learning outcome

Assignment: Portfolio

1. How will students submit the portfolio (in form)?
2. How often will student submit for feedback?
3. How often should they work on the portfolio?
4. What kind of content will students contribut to their portfolio?
5. What kinds of prompts will guide their work?

Assignment Outline for Specifications or Competency-Based Grading

Motivation (Why are we doing this?)

```
- What is the purpose of this assignment?
- Briefly, what will students be doing?
```

Pre-Req Info

```
- What is the assumed or expected prerequisite knowledge or background students need to complete this assignment?
- What are some resources for students to refresh or review said prerequisite knowledge or background?
```

Background Info

```
- What course or assignment specific information is necessary for students to successfully complete this assignment?
- Are there any examples that may be helpful?
- Are there any additional resources that may be helpful?
```

Your Task

```
- What exactly should students be doing? Bullet points and numbered steps help make assignmetns more clear.
```

Requirements

```
- What should the finished assignment do/have/look like?
- If using competency-based grading, requirements should be outlined in increasing order of difficulty
```

Handing in

```
- How will students submit their work?
```

Grade Breakdown

```
- If using specifications grading, what is the minimum number of requirements needed to pass the assignment?
- If using competency-based grading, what are the specific requirements needed to demonstrate various levels of knowledge?
```

Frequently Asked Questions (FAQs)

Click a question or concern to expand its answer or suggestion below.

Getting Started Questions

Can these principles apply to my subject area?

Will applying these principles make grading take more time?

Are these practices evidence-based?

How long does it take for students to understand and get used to alternative grading schemes?

How have students adapted to these new systems? I'm concerned that all of these different grading systems will make the students confused/anxious.

How much pushback should I expect from students?

Logistics Questions

My LMS doesn't support alternative grading structures.

My LMS doesn't support peer-grading.

How can I apply these principles while using automated grading?

When resubmissions are allowed, how much feedback fo you provide?

Does allowing resubmissions or flexible deadlines mean more grading? Are there a lot of last minute submissions?

What is the trade off of extra flexibility and allowing continual re-submissions. Will this set a precedent that students expect all deadlines and work can be flexible or changed? Do you think not having deadlines or having flexible deadlines negatively impacts them in the real world where they need to meet deadlines in their jobs?

Additional Resources

By Sarah M Brown and Victoria Chávez