3013-Algorithms

Roster

4443 - Class Roster

General Course Info

- **Days:** MW 2:00 p.m. - 3:20 p.m.
- **Location:** BO 209
- **Semester:** Tuesday January 16th - Friday May 3rd
- **Holidays:**
  - **Spring Break** Monday March 11th - Friday March 15th
  - **Easter** Monday March 27th - Friday March 29th
- **Last Day for "W":** Monday March 25th
- **Last Day of Class:** Friday May 3rd
- **Final Exam:** Wednesday, May 8th @ 5:45pm – 7:45pm (same room)

Office Hours

Dr. Terry Griffin
Spring 2024
phone: (940) 397-4439 email:terry.griffin@msutexas.edu

	MONDAY	WEDNESDAY	FRIDAY
8:00			
8:30			
9:00			
9:30			
10:00			
10:30			
11:00			
11:30			
12:00	Lunch	Lunch	Lunch
12:30			
13:00	Office	Office	
13:30			
14:00	CMPS 3013	CMPS 3013	Research Group
14:30	BO 209	BO 209	
15:00	Start 2:00	Start 2:00	
15:30	End: 3:20	End: 3:20	
16:00	Office	Office	
16:30			
17:00			
17:30			

	TUESDAY	THURSDAY
8:00		
8:30		
9:00		
9:30		
10:00		
10:30		
11:00	CMPS 4443	CMPS 4443
11:30	BO 109	BO 109
12:00	Start: 11:00	Start: 11:00
12:30	End: 12:20	End: 12:20
13:00	Lunch	Lunch
13:30		
14:00		
14:30		
15:00	Office	Office
15:30		
16:00	CMPS 4443	CMPS 4443
16:30	BO 109	BO 109
17:00	Start: 16:00	Start: 16:00
17:30	End: 17:20	End: 17:20

Office hours are as scheduled or by appointment only

Resources

Here are some open source books for the course. I hope you guys appreciate the amount of effort it takes to put material together and then put it on the internet for free.

- Discrete Structures for Computer Science: Counting, Recursion, and Probability
  - Thanks To: Michael Smid

- Open Data Structures
  - Code available HERE
  - Thanks To: Pat Morin
- Algorithms
  - Thanks To: Jeff Erickson
- Wikipedia Collection of Data Structures

Assumed

- This course assumes you know what `array based data structures` and `list based data structures` are.
- For example you should be able to write (from scratch) an `array based stack` and `queue` along with a `list based stack` and `queue`. If you cannot, go study.
- You should have a general understanding of recursive functions.
- You should have a general understanding on graph structures more specifically be able to write and traverse a basic Binary Search Tree (BST).
- Basic OOP skills. Mainly encapsulation and implementation hiding, in other words packaging a data structure with the methods to manipulate that data structure.

Note About This Courses

I will try to go over the list of topics (see below) in the order they are listed, and I have a path that I like to follow when introducing these topics. However, I approach each course with the hopes that student interaction and feedback will steer the course in a direction that may not be what was previously planned. This is stressful for some of my more "organized" students, and I can appreciate their angst. So I always provide a study guide before each exam to ensure everyone is on the same page with the topics I expect you to study. The study guide could have actual test type questions, small programs to implement, or vocabulary and topics.

Algorithm Categories

- Backtracking
- Brute Force
- Dynamic Programming
- Greedy Algorithms
- Recursion

Topics List

As I stated above, we will try to stay on task, but let me give you an example of why things can't always be taught sequentially. There are always multiple ways to implement every data structure. For example there is a data structure called a **priority queue**. It can be implemented in many ways, some better than others. Some of these won't make sense right now, but you will get the idea. Here are a few:

1. Use an **array**, and order the items in the array using the "priority" value.
2. Use a **singly linked list**, and order the nodes using the "priority" value.
3. Use an **array based binary tree**, called a **binary heap** and using the "priority" value to order the heap.

4. Use a **doubly circular linked list**, and follow the algorithm known as a **Fibonacci Heap** to keep the items in order.

So ultimately based on class input and questions that come up in discussion, I may jump around a bit so an explanation will make sense. Remember though ... *I make study guides for each test*!!

- [ ] Array Based vs List Based Structures
- [ ] Array Based Implementations
- [ ] Complexity
  - [ ] Introduction
  - [ ] Will be discussed with each data structure
- [ ] Linked List Types
  - [ ] Singly Linked List
  - [ ] Doubly Linked List
  - [ ] Circular List
- [ ] Stack, Queue, Priority Queue, Deque
- [ ] Array Based Binary Search
- [ ] Binary Tree's
  - [ ] Components
  - [ ] Array Based
- [ ] Binary Heap (Array Based)
- [ ] Fibonacci Heap (**Possibly**)
- [ ] Binary Tree Implementation (List Based)
- [ ] Trie
- [ ] Balanced Tree's
  - [ ] AVL
  - [ ] Red Black (**Possibly**)
- [ ] Hash Tables
- [ ] Graphs
  - [ ] Array Based and List Based Implementations
  - [ ] Basic Graph Algorithms
  - [ ] DFS (Depth-First Search)
  - [ ] BFS (Breadth-First Search)
  - [ ] Minimum Spanning Trees
    - [ ] Prim's Algorithm (Minimum Spanning Tree)
    - [ ] Kruskal's Algorithm (Minimum Spanning Tree)
  - [ ] Shortest Path
    - [ ] Dijkstra's Algorithm (One Way Shortest Path)
    - [ ] A-Star Algorithm (**Possibly**)
- [ ] Sorting:
  - [ ] O(n^2)
    - [ ] Bubble Sort
    - [ ] Selection Sort
    - [ ] Insertion Sort
  - [ ] O(n lg n)
    - [ ] Merge Sort
    - [ ] Quick Sort

    - [ ] Other
        - [ ] Counting Sort
        - [ ] Radix Sort

Resources

- http://opendatastructures.org/ods-cpp/
- https://github.com/ippeb/ACM-ICPC

Grading

Grading within the recent past has become a fluid / difficult endeavor. Projects (programs) do not have the same integrity that they once did in assessing a students knowledge. Exams, like always, evaluate a students ability to memorize, cram information, or cheat I dare say, all without really understanding the topics. ***So I will be going for a more hands on approach by using oral examinations in conjunction with code reviews.*** This means I will sit down with everyone at least once, on a project of my choosing to discuss with them the code they submitted for grading. This way, whether its a group project or individual project, I can assess a general understanding from each student for that project. I wish I had time to do this for every assignment, but it wouldn't be feasible from a time approach.

The grade scale below is not complete, but will change as our class moves forward. I will use a democratic approach to determining a final grading scale ensuring the majority of the class agrees with the weights for each category. I will reserve the right to veto ridiculous organized labor's ideas (aka class getting together and all agreeing on something ridiculous), but will in good faith try to listen to the class within reason.

Categories	Portion of Course	:::	Letter Grade	Grade Range
Exams	TBD	:::	A	90-100
Project(s)	TBD	:::	B	80-89
Final	TBD	:::	C	70-79
Github	TBD	:::	D	60-69
Participation	TBD	:::	F	below 60
Presentation	TBD	:::		
Code Review	TBD	:::		

- **Exams**:
    - Not sure if I want this to be a pure projects based course, or supplement projects with exams. This will depend on class participation and my opinion on how the class as a whole is absorbing the information.
- **Projects**:
    - These are programming assignments. I do not put a lot of weight into projects (programs) as of late. Starting with Google, then onto Stack Overflow, and now with Chat GPT it's very hard

to determine ones individual work. So I will be assessing your work with a different set of metrics (Aka Presentations, and Code Reviews).

- **Final**:
    - TBD. Not sure how I want to approach this yet.
- **Github**:
    - This is a portfolio of your work. It somewhat defines you as a computer scientist. An ugly portfolio will not generate interest in you by companies looking to hire. A good well organized portfolio with strong and detailed explanations of projects will help you get noticed. Oh, and get you a good grade for this component.
- **Participation**:
    - What is participation? I know many individuals in class may be shy, especially given the personality type that dominates our major (including me when I was your age). So how can you `participate`?
        1. The best way to participate is simply to provide comments and or questions during class. But as stated above, I know not everyone is comfortable doing this.
        2. So, you can post a question pertaining to the previous class or an upcoming class on Slack. This can be done via DM to me, or on our course channel. **Questions drive my lecture, so they are always welcome**.
        3. Try and include yourself in conversations that start up on our Slack channel. If you can answer someones question, or help clarify a requirement, please do so. This also gives me the opportunity to jump in and fix something that may be ambiguous or confusing in my assignment.
        4. Provide links of cool tutorials, libraries, or assets to the class. Again, if you're unsure, DM me, and I will help filter. But I will also give you credit for cool finds.
        5. Ask questions about my project descriptions on our course channel or in a DM. These questions are extremely helpful! Every semester I pretty much create new projects, and this leads to ambiguities or unattainable goals (rarely). Simply asking me to clarify something on a study guide or project description that you find confusing helps me tremendously. Again, DM me directly, or post on course channel.
    - Any communication I receive from ya'll in a DM will be redirected by me to the course channel. I will leave your name out of my comment if explicitly asked, but good questions deserve credit, so I usually give credit where it is due. I will **never** out someone or try and embarrass someone based on private conversations between myself and that person. Unless you get embarrassed by praise and accolades, since that's the only way I mention a name in the public channel based on a private conversation (DM).
- **Presentation(s)**:
    - Presentations may be individual or with a group (instructor preference).
    - The presentation should show and discuss a specified project. Your discussion will include:
        - The positives, and negatives of your project.
            - Positives may include something cool, above and beyond the requirements, or something you discovered that was rememberable.

            - Negatives may include the troubles getting particular requirements to work or particular components you had trouble getting to work. Basically, the portions of the project you had the worst time with.
        - Always ask yourself about code efficiency. Did it come into play? Did you think about it at any point in fulfilling a specific requirement? Was efficiency not an issue at all?
        - What was the funnest part of the project?
        - What was the most difficult part of the project?
    - Showing code specific portions of your code for the above items is a good way to help you walk your viewers through each of the points mentioned above.
    -

- **Code Review**:

    - As mentioned above:
    - I will choose a random project and have individuals describe in detail chunks of code chosen by me.
    - This may be in front of class, or simply with only me.
    - My goal is to find out what you actually know about your project code.

Teaching Philosophy

I approach each course with the mindset that every class has a different set of students and not every lecture or list of topics should be given in a "lock step" manner tied to a calendar. I try and incorporate current industry topics, student questions, and student interests into my everyday lectures. I will always cover the core content of the course, but it just may not be in the same order, or using the same canned examples for each topic every time. A single question may send our lectures off on a path not previously planned.

I call this my "problem based approach" to teaching. Usually a question is dealing with a specific problem, so I tailor my lectures to incorporate topics necessary to solve the problem. It also means topics do not get delivered in the same order every semester. Of course, I guide the solution we use to keep in line with content commensurate with overall course objectives.

This method of content delivery is not for everyone. But based on the vast majority of course evaluations most students do enjoy my lecture style and content delivery. Having said this, I know my methods are not perfect and not every student responds positively. To alleviate distress for those students I create study guides for each exam. This way no matter what order content is delivered, they have a concrete list of topics whereby the exam is a subset of those topics. In fact, I give the students more than just a list of topics to study, I also provide example exam questions to that can be answered as part of the study process.

How to Succeed

I encourage every student to firstly just **go to class**! Beyond that, attempt to participate in class discussions and also ask questions in class. Believe me, classes that have many student questions with a subsequent discussion, seem do much better in understand a topic as a class.

More importantly you should interact with your fellow students outside of class as well. Start a study group, post links on Slack, ask questions on Slack! Stay involved with your classmates. Also, if you see posts on slack, respond to these posts either with a text response or an emoticon reaction to the post. No one likes to post something, and then feel as no one has read it.

If you need help in understanding a topic, you need to message me immediately so I can either text you a response, or zoom with you to get things cleared up. And yes I will zoom late into the evening if that is necessary. I'm not saying I'm at your beckon call, but I try to make myself very available.

Lastly, this isn't totally about your own success, but it helps me succeed. Asking me questions with direct message or on the course channel is a huge help! I can turn that question into (what the military calls) an "overhead correction", meaning I can clarify something to the entire class. It does not mean I will use your name (unless you publicly post it, then that is on you 😀), but it does give me the opportunity to clarify things to the entire class, since most questions tend to be what the majority of students in the class are thinking.

My View on Cheating / Plagiarism

- Most plagiarizing, when it comes to programming, happens for two reasons:
    - 1. You don't have a clue how to solve the problem, so you get a friend or the internet to help.

> This can be ok, if you cite your resource, and only find small snippets of code to work into your own solution.

    - 2. You didn't start early enough, and you're desperate to get something working the night before it's due, so you get a friend or the internet to help.

> This is never ok.

- Both are easy to fix.
    - 1. Come ask me to explain. I promise you're not the only one who is confused.
    - 2. Start early. Then when you get stuck, you can ask for help the right way!

Presentations

- Presentations are a major component of your course work. The ability to discuss complex topics in front a group of your peers is an important skill to have.
- Depending on the course, and the size of the class, you may have many presentations, or just one presentation.
- The quality of a presentations that accompanies a programming project is highly coupled with the quality of your project. A poor project makes it hard to give a proper presentation on a project in which you did not complete.
- On the other hand, an excellent project doesn't ensure a great presentation either.
- **Preparation is key**, and I am **ALWAYS** available for help with presentations.
- I will give specific requirements for each presentation since each project may vary greatly, but in general project presentations in my course should follow a basic outline:
    - Project description (if necessary)
    - A logical progression of your steps in implementing the project. Make sure to include:

        - Pitfalls (any confusing components that gave problems)
        - Highlights (any good solutions or components you are proud of)
        - Summarize the results or final product whether it be the completed features of the project or the results of any data you processed.
- Be prepared! Sometimes showing your project seems easy since you spent many hours writing it and have a very deep understanding of it, but this does not translate to a good presentation.
- A good presentation is well thought out and practiced.
- Side Notes:
    - A well thought out presentation allows you to hide flaws or unfortunate "features" that you may not want anyone to know about.
    - I am also much less inclined to ask pointed questions if you have a well thought out and thorough presentation.

Miscellaneous

Some of these points are duplicated in other places in this document. It's ok.

- All students need a Github account
- All programs need to be turned in and running to pass the course
- General Assignment Rules:
- Due dates and times are as listed on assignment and can change with prior notice to class, and always in your favor (aka more time).
- Formatting of programs is important, and will be graded accordingly.
- You name is required on ALL documents uploaded or turned in. A handwritten name is not acceptable.
- For any assignment, you will create a folder and include all documents created by you within this folder for submission. This includes programs, input / output files, readme's, documentation, etc. This folder will subsequently be uploaded to Github in the repo you created for the course.
- Attending class is one of the primary keys to doing well in this class. Students may be dropped for excessive absences. There is no distinction made between excused and unexcused.
- Make-up exams are not given. If I see fit, then I will replace a missed exam with your final exam test grade (but this is optional to instructor based on circumstances, attendance, participation, etc.).
- Late work will be accepted on a case by case basis. Late penalty is 15 points (out of 100) for initial lateness and 1 half a letter grade (5 points) for every class period until the total reduced is 50 (half credit). Extremely late work is totally at the instructors discretion on whether it will be accepted or not.
- Programs containing syntax errors are unacceptable and will be returned without grading (your programs must work).
- Periodically homework assignments will be taken up and graded. It is the student's responsibility to keep up with assignments and to ask questions over the assigned work, even if absent. All homework assignments are due at the specified time that may or may not be in conjunction with a class day. All assignments / homeworks will be uploaded via Github.

Official Course and Department Policies

Attendance Policy

I don't have an attendance policy. However, I do track students attendance and will submit a drop request to the dean of students for students that have excessive absences. I won't define "excessive" but will give you a warning before I submit the actual drop request. To alleviate issues, simply communicate with me about any issues or problems you are having with getting to class, and I will help in any way that I can.

Behavior in the classroom

Students are to assist in maintaining a classroom environment that is conducive to learning. This means that the presence of electronic devices other than your calculator are not to be seen, heard, or implied, ever. Questions are encouraged and discussion is acceptable, provided it is pertinent and does not distract from the lesson.

Make Up Work/Exams/Quizzes:

- For planned excused absences: Exam may be taken early by prior arrangement.
- For unplanned documented absences: I may replace your exam grade with your final exam grade. I reserve the right to make that decision.
- For unplanned undocumented absences: Zero on the exam

Late Work

Late programs / homeworks will be accepted with an initial **15 point** reduction and then a **5 point **deduction class day. ***No late programs for last programming assignment.***

Computer Requirements: Taking this class requires you to have access to a computer (with Internet access) to access online course material. ***Personal computer technical difficulties will not be considered a reason for extra time to submit assignments, tests, or online discussion postings.*** Computers are available on campus in various areas of the buildings, as well as the in the library. Contact your instructor immediately upon having computer trouble. If you have technical difficulties in the course, there is also a student helpdesk available to you. The university cannot work directly on student computers due to both liability and resource limitations, however they are able to help you get connected to our online services. For help, log into `D2L`

Policy on Testing Process

The Department of Computer Science has adopted the following policy related to testing.

- All bags, purses, electronics (turned off), books, etc. will be placed in the front of the room during exams, or in an area designated by the instructor.
- Unless otherwise announced by the instructor, nothing is allowed on the desk but pen/pencil/eraser and test papers.
- A student who leaves the room during an exam must turn in the test and will not be allowed to return.

Policy on Programs

- Tests *will* have questions covering out-of-class assignments. Know the material!
- Students will be invited to orally answer questions regarding their assignments in my office and failure to answer those questions correctly will result in deductions from their grades. (Every student can expect to be invited 1-2 times during the semester to answer questions.)

Computer Science Tutoring

Tutoring is available in ***Bolin Room 119 & the Office of Tutoring and Academic Support Programs (TASP)*** in Moffett Library. A tutor may assist with programs and homework for computer science classes. The tutor will not do your work.

Academic Misconduct Policy & Procedures

Cheating, collusion, and plagiarism (the act of using source material of other persons, either published or unpublished, without following the accepted techniques of crediting, or the submission for credit of work not the individual's to whom credit is given). The Department of Computer Science has adopted the following policy related to cheating (academic misconduct). The policy will be applied to all instances of cheating on assignments and exams as determined by the instructor of the course. (See below for link to MSU definitions.)**

- 1st instance of cheating in a course: The student will be assigned a non-replaceable grade of zero for the assignment, project or exam. *If the final grade in the course, does not result in a one letter grade reduction, the student will receive a one letter grade reduction in course.*
- 2nd instance of cheating in a course: The student will receive a grade of F in course & immediately be removed from course.
- All instances of cheating will be reported to the Department Chair and, in the case of graduate students, to the Department Graduate Coordinator.

Note: Letting a student look at your work is collusion and is academic misconduct!

See Also: MSU Student Handbook: Appendix E: Academic Misconduct Policy & Procedures *https://msutexas.edu/student-life/_assets/files/handbook.pdf*.

Recording of Class Lectures

Permission must be requested in writing and obtained from the instructor before recording of class lectures. If permission is granted, the recording may only be used by the student making the recording. Recordings (or any class materials) may NOT be posted on any internet source without written permission of the instructor. Failure to adhere to the policy may result in removal from the course with a grade of F or other appropriate punishment.

University Policies and Procedures

Student with Disabilities

Any student who, because of a disability, may require special arrangements in order to meet the course requirements should contact the instructor as soon as possible. Students should present appropriate verification from Disability Support Office during the instructor's office hours. Please note instructors are not allowed to provide classroom accommodations to a student until appropriate verification has been provided. For additional information, contact the Disability Support Office in Clark Student Center 168 - Phone: (940) 397-4140

Policy on Concealed Handguns on Campus

Senate Bill 11 passed by the 84th Texas Legislature allows licensed handgun holders to carry *concealed* handguns on campus, effective August 1, 2016. Please note, open carry of handguns, whether licensed or not, and the carrying of all other firearms, whether open or concealed, are prohibited on campus. Areas excluded from concealed carry are appropriately marked, in accordance with state law. For more information regarding campus carry, please refer to the University's webpage at MSU Campus Carry Policy *https://msutexas.edu/campus-carry/rules-policies### *. If you have questions or concerns, please contact MSU Chief of Police Steven Callarman at Steven.callarman@msutexas.edu.

Midterm Progress Report

In order to help students keep track of their progress toward course objectives, the instructor for this class will provide a Midterm Progress Report for all students in the course through each student's MSU Portal account. Midterm grades will not be reported on the students' transcript; nor will they be calculated in the cumulative GPA. They simply give students an idea of where they stand at the midpoint of the semester. Students earning below a C at the midway point should a) schedule a meeting with the professor and b) Seek out tutoring.