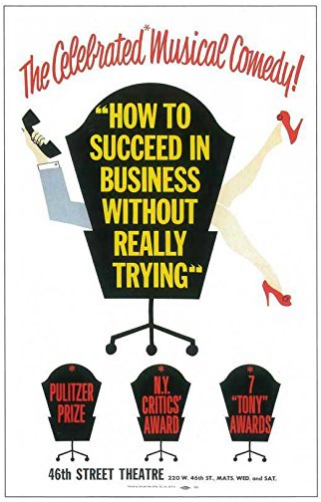
CMPU 240 · Fall 2022

How to Succeed in CMPU 240 by Really Trying

You'll see a lot of detailed information – definitions, proofs, examples, applications – that support the big ideas of this course. To get the most out of it, here are some suggestions for approaching the course material.



Close – but try harder.

Take notes in class

While the slides from each class are posted to the course website for your reference, it still helps to take notes. Rather than write down what's on the slides or try to write down everything I say, put the concepts into your own words. You may find it especially helpful to draw diagrams showing how different ideas relate.

Also, pay attention to the questions other students ask and the answers to them. These won't be in the slides, but they may be the same questions you're wondering.

Stay on top of the material

The concepts in this course build on each other. If there's something you don't understand, you need to make an effort to clarify it when it comes up.

I recommend you read the required material after the class where it is introduced. This way, you are reinforcing what you already understand with another explanation.

Focus on the main concepts and not just the details. For example, don't just memorize the steps to convert a DFA to an NFA, but understand how the conversion works in principle. The steps then become easy to remember.

It's tempting to focus all your time on the current assignment, but previous misunderstandings will come back to haunt you! This is the primary reason you're being asked to correct your assignments after you submit them – so you need to think about what you did right and what could be improved.

Start assignments early

If you leave assignments until the night before, you'll struggle just to finish before the deadline, which means you'll be working harder and learning less!

Start the assignments several days before they are due. The problems really benefit from going away and coming back later when you are stuck. And if you start early, you'll have time to get help if you realize there's something you don't understand.

Know when to struggle and when to stop

It's important to make a serious effort to understand the material and to work through the assigned problems. Always asking for help is an effective way to "complete" your assignments, but it's a terrible way to learn from them, and it sets you up for failure on the exams.

On the other hand, there's no prize for spending hours and hours hitting your head against a problem. If you've made a sincere effort to answer an exercise, reviewed the lectures and textbook, and you're still not sure how to proceed, ask for help!

It's up to you to learn to calibrate the right amount of perseverance – when to keep struggling and when to get help.

Learn together

Computer scientists have an undeserved reputation for being loners. In fact, computer science is a highly collaborative field, and computer scientists love to explore new ideas together. Remember that *you're not alone in this course*.

It's a good idea to study together, since explaining concepts to peers and hearing their explanations can help to clarify your understanding. You're also allowed to do assignments in pairs, and I recommend you try it, but focus on learning the material, not on completing the assignment as quickly as possible.

I recommend that you attempt all of the problems on your own *before* coming together as a pair. Do *not* divide up the problems. The assignments provide practice at different skills; some problems are focused on terms and definitions, some are explorations of pure theory, and others explore applications. If you take the time to work through them and really think about the material, you'll come away with a nuanced understanding of the concepts from the course and

how to apply them. On the other hand, if you split the work up, your understanding will be incomplete.

You can do this!

This course requires mathematical thinking and problem-solving that can feel quite different than what you do when you design computer programs. Many students – including many Computer Science majors – tell themselves that they're just not good at math, so there's no point trying. Don't fall into this trap!

For the perspective of a self-described coder who was convinced he was "bad at math" and learned that this didn't need to be the case, I recommend reading the article "How I faced my fears and learned to be good at math". Theoretical computer science is something that can only be learned with practice.

Sarah Andersen

Acknowledgments

This is based on guides by Nancy Ide (Vassar College) and Keith Schwarz (Stanford University).