Computer Science 167 and 169

Course Policies
Spring 2017

IMPORTANT: You will receive a link to a Google form where you must affirm that you will abide by the following collaboration policy.

Collaboration Policy

Collaboration policies in CS167 and CS169, are generally loose, in order to stimulate a better learning environment. Without going outside the basic Academic Code (see Basic Policy, p. 4 of "Academic Code and Non-Academic Disciplinary System") we are providing the following set of guidelines.

The basic premise is that you should do your own thinking, your own design, and your own coding. You're allowed to talk to other students about the content of the lectures and of the textbook and about high-level concepts in general. You can answer questions from other students about packages used for assignments, as long as the problem is a narrow one and not one that helps in the problem-solving process at large. You may also help another student with coding issues that are strictly language-related. Finally, you can assist another student with debugging if they are stuck with a specific low-level problem that has been impeding progress on the work.

On a general level, what is not allowed is that you let yourself be led by another student to the extent that your task becomes significantly less challenging because of your discussion with them. More specifically, you should do your own problem solving, program design and decomposition, and design your own data structures. In conversation with other students, be sure not to venture into design and coding specifics, and especially never sit down to discuss an assignment with someone else before you've analyzed the problem in depth on your own.

The most blatant violation that can occur is code-copying; this absolutely will not be tolerated. We reserve the right to do a "wire-pull test" (i.e., ask you to explain your program) and to use all of the tools at our disposal to compare your code to that of other students (including assignments from years past). In a similar vein, make sure that all of your coursework on the filesystem has the proper permissions so that other students cannot view and potentially copy your work. See `chmod(1)` or ask a consultant for help if you don't know how to go about this. Failure to do this can potentially be viewed as criminal negligence.

Similar guidelines hold for written homework assignments. You may work in groups in the process of solving the problems, but in all circumstances, the written answer must be your work. You must completely understand the answers you give, and we reserve the same "wire-pull test" rights as on programs.

If you are ever in doubt about the legality of your actions, be sure to clear them with a TA, even if it is after the event has occurred. If Professor Doeppner confronts a student about a suspected academic code violation, an answer of "I didn't know that this was wrong" is not likely to find much sympathy.

Again, note that you are expected to always approach a problem initially on your own and seriously attempt to find a solution. You are honor-bound to preserve your independence of thinking. And remember that the TAs should always be your first resource when you have a question or problem.

TA hours are intended for students to use as a resource for getting help with assignments (both programs and homeworks). It is expected that, before coming to a TA for help, you have made a significant attempt on your own to resolve your problems. For homeworks, this means that you have thoroughly considered the question and possible solutions and are, perhaps, unclear as to the nature of the question or some of the concepts involved. For programs, in order to receive help, you must have made a serious attempt to trace your bug to its source, or at least isolate its occurrence to a few specific scenarios (most likely using `gdb`). Simply stating "I have a bug" will result in no help whatsoever from your TA other than the friendly suggestion that you try using `gdb`. If you think you have found a bug in any of the TA-supplied code, please isolate the bug and provide the TAs with a few explicit scenarios in which it occurs so that we can reproduce

and subsequently fix the problem.

That being said, asking TAs for help during times not explicitly designed as their hours is forbidden[1].

Finally, please do not post course projects publicly to sites like Github or other repository-hosting sites. This could be (and has been) construed as an academic code violation. Github allows you to create private repositories for free with student accounts.

Examples

The following are examples of collaboration, and our interpretation of how the policy applies to them. Feel free to refer to them for clarification.

- **Archita:** I have a bug where typing in the command line doesn't work!
  Egor: Oh, I had a similar bug! Did you check your Drivers code?
  Helping a peer locate their bug in general **is acceptable**
- **Steven:** I haved a bug where typing in the command line doesn't work!
  Isaac: Oh, I had a similar bug! Did you check your Drivers code? I had a bug in `n_tty_receive_char()`!
  Suggesting a classmate check out a particular function **is acceptable**.
- **Kyle:** I have a bug where typing in the command line doesn't work!
  Ian: Oh, I had a similar bug! Did you check your Drivers code? It sounds like you have a buffer overflow.
  Suggesting a concept that may be causing a bug **is acceptable**.
- **Archita:** I have a bug where typing in the command line doesn't work! I discovered that it's caused by a buffer overflow.
  Steven: Do you have an off-by-one error when copying into the buffer?
  Once a student has discovered the cause of a bug, suggesting general things to check for, that are not specific to this bug, **is acceptable**.
- **Egor:** I have a bug where typing in the command line doesn't work!
  Isaac: Oh, I had a similar bug! Did you check your Drivers code? I had a bug where `n_tty_receive_char()` was returning 3 instead of 4.
  Here, Ian makes Kyle's job significantly easier by telling her not only where the bug might be, but what it is. This is **not acceptable**.
- **Isaac:** I have a bug where typing in the command line doesn't work!
  Ian: Have you tried using gdb to see if you're reading the characters correctly?
  Suggesting debugging strategies **is acceptable**.
- **Archita:** Can you explain what a mutex does?
  Isaac: Sure! It (blah blah blah).
  Explaining a programming construct and/or class concept **is acceptable**.
- **Egor:** Can you explain what `proc_kill_all` does?
  Kyle: Sure! It terminates all processes, except for the idle process and its children.
  Explaining and clarifying the documentation, including explaining what a function is supposed to do and its purpose, without discussing specifically how to implement it, **is acceptable**.
- **Archita:** Can you explain what `proc_kill_all` does?
  Kyle: Sure! First, it does (this). Then, it does (this). Then, it does (this). ...
  Unless the comments for the function are equally as detailed, students should not give each other step-by-step instructions for how to implement functions.

[1] Note that, because of the nature of CS169, often mentor TAs allow their students to contact them with questions when they're off hours.

- **Egor:** Why do we have multiple processes again?
  Ian: blah blah blah concurrency blah blah blah.
  Explaining course concepts at a high level to other students **is acceptable**.

- **Steven:** I don't really understand how system calls work.
  Archita: Oh, it talks about that on page (something) of the textbook.
  Helping classmates locate information in course materials **is acceptable**.

- **Isaac:** I don't really get question 2 on the homework. What is a system call again?
  Egor: A system call is (blah blah blah).
  Explaining course concepts used on an assignment to classmates **is acceptable**.

- **Kyle:** I think I'm done with Procs!
  Steven: Are you sure you pass the case where multiple threads try to lock the same mutex?
  Suggesting edge cases and general things to test for **is acceptable**.

- **Ian:** I think I'm done with Procs!
  Isaac: Are you sure you pass the case where multiple threads try to lock the same mutex? I failed that initially because I was checking for waiting threads in `lock` instead of `unlock`.
  Explaining specific causes for tests failing is **not acceptable**.

- **Archita:** How are you testing Procs?
  Ian: Generally, I'm trying to create a lot of processes and having them do things that require all the functionality I implemented.
  Describing general testing strategies **is acceptable**.

- **Egor:** I haven't gotten a chance to read the handout yet. Could you give me an overview about how multiple processes work?
  Archita: Sure! Each process runs and executes code until it goes to sleep, and then if there are other processes waiting to run, they run until they go to sleep, and so on.
  Explaining course concepts to other students relavent to projects **is acceptable**.

- **Steven:** I haven't gotten a chance to read the handout yet. Could you give me an overview about how multiple processes work?
  Egor: Sure! In Weenix, each process has a single thread. When you create a new process, you create a new thread. ...
  Students should read the handout before asking other students about the specific design of programs.

- **Isaac:** I can't find the spot where this process' process id is set!
  Steven: Have you tried using gdb to watch the memory address? Here, let me show you how to do that.
  Showing other students how to use debugging tools **is acceptable**.

Piazza Guidelines

Piazza is a place for sharing questions about the assignments in this course. Feel free to ask questions about the homeworks and projects, and in studying for exams. We offer the following guidelines for effective use of Piazza:

- Please tag your posts with the appropriate tag when you create your post.

- As in all interactions with other students in this course, you must adhere to the collaboration policy when interacting over Piazza.