The following document is written by Matthew Tracy

This post is your go-to guide on how to post questions. If you follow this guide, not only can it be guaranteed that you get your issues resolved both swiftly and concisely, you will also create questions that can benefit your fellow students.

**Private Posts**

As a general rule, there are only two circumstances in which a private post is appropriate:

1) A teaching assistant or Professor Sacks specifically requests that you create a private post.

2) You have a question regarding a personal matter unrelated to assignments in progress (i.e. issues with grades, our late policy, unsatisfied with a lab session or TA, etc).

If your post is not prompted by one of these two circumstances, please do not make a private post. If you do not want your peers knowing that you are asking a question, you are able to set a preference when writing your post so that your name is anonymous to your peers, while still allowing you to post in the public domain.

As for content in a private post. **DO NOT POST ANY CODE!!!!!**

Regardless of what you have heard before, do not post your code in a private post. We have access to your code, and there is no reason for it to be posted on Piazza, even in a private post. If another student borrows your computer and sees your private post, your code has been compromised, and if they copy what they see you may find yourself flagged for cheating (we'll get back to that in a bit).

What can be included in private posts regarding your test cases (how many you include, if you may include them, etc.) is at the discretion of the instructor who requests you create a private post for your issue.

**Public Posts**

Posting questions publicly can be intimidating: we have been unclear to what we want, and no one likes when a TA has to censor part of your question, sometimes rendering it useless to other viewers. In some cases, we even have to delete the question entirely. That is why learning how to write questions on Piazza is arguably the most important skill in this class. The problems that most of you (and when I mean most, I mean 99%) experience with your programs are not unique—if you are confused, I can guarantee that at least 50 students feel the same way. With lab 1, there have been dozens of redundancies in posts, which is why it is paramount we write questions in a way such that students can identify their issue so that they do not have to write their own question, and also so that they know how to resolve the issue that you were having which prompted your post. To write a good Piazza post, you first have to make sure the same question hasn't already been asked so...

~SEARCH FOR YOUR QUESTION BEFORE ASKING IT~ There's a search bar up there

**Writing Titles**

In order to write an effective title, there are two goals to keep in mind:

1) Other students who look at the title should have a general idea of where you have your issue.
Posting a question titled “I don’t know what is wrong” or “Please help” gives someone who is glancing through the index of questions no idea what your question is about. Summarizing your issue in a single line is vital to being able to identifying the problem. If you cannot identify your problem, you may want to run a few more tests before you post. Otherwise, it does not look like you spent enough time trying to debug on your own. A general rule of thumb is that you should post your question if you cannot resolve your issue after an hour of debugging that specific problem.

2) While your question should summarize your issue, it should not be too specific to the point that other students cannot relate to the problem you are having. It should be ambiguous enough that other students can identify related circumstances and know that it is not unique to you (because trust me, it isn’t), but specific enough that everyone who reads your title knows what you are talking about. Posting a question titled “Test Cases 1, 3, 5, 11…” is not appropriate. The only time that a question referencing specific test cases is appropriate is when you are continually failing a single case. If you are failing multiple cases, then your issue is most likely not a corner case, and more general in nature. If you follow these rules, then you can make a title that both students and instructors can quickly identify its subject matter and know if it is relevant to issues they are also experiencing.

Contents of a Piazza Post

Your contents of a Piazza post are just as important as its title. When you write a question, it is necessary that you do three things:

1) Identify the problem. You may not know exactly what is going on (otherwise why post), but saying “I don’t know” is not information we can use.

2) State your diagnostics process. No. This does not mean post your test cases The art of writing a post in Piazza is not to make your peers and the instructors serve as an enhanced version of the autograder. The purpose of piazza is to act as a discussion forum through which the class can talk about how a program is supposed to function and areas in which one might need clarification. Let me show you an example: A student can say “I’m having an issue and I have no idea what is going on, here is my input: cpp yolo but for some reason this is my output: #cpp #yolo&*#@# OR alternatively it can be asked with the following format: “For some reason when I am hashing words in my program, I am receiving garbage values at the end of my output. Why might this be occurring?” With both of these question formats, the exact same thing is asked. However, the former question risks being censored because it posts a test case. In general, you should never post a specific test case. If we allowed this to happen, then students could simply sift through Piazza, collect test cases, and debug until they had a perfect program without learning how to find these cases themselves. By learning to describe the problem, the question goes from being a question of “tell me what is wrong” to “tell me how to look for an answer”. Having questions that ask for direction are far more valuable than those which ask for answers, and in the long run will help you grow as a programmer.

3) Make sure that the solution is visible when you have resolved your problem. On far too many occasions students ask their question, identify and fix the problem, but never summarize how they identified their issue or how it was resolved. If a post does not state its solution, it is worthless. Keep in mind that Piazza is not only meant to serve you, but also your peers. In addition, instructors may also be interested in how you found and fixed your problem ;). Once you have found the solution, either update the student answer, write your solution at the bottom of your question, or at the end of a follow up (allowing others see the process of diagnosis) so that students can gain insight from your experience. By crafting your questions in a format that is not only thorough but also avoids giving
away cases students can extract without reading the discussion that follows.

**Things to Remember**

While Piazza can be frustrating, there are a few things to always keep in mind (I’ll make it short, if you've gotten this far, thanks for reading so much ;) ):

• Write posts that are not only for yourself, but also for others. There is almost always someone else with the same question, therefore always write in a way that does not imply that your question is an anomaly unique to your program. It is not.

• In follow ups and student answers, use clarity. Sometimes it is necessary to request a student give better insight to their issue to receive assistance, however, one-line answers are almost never enough to constitute a full-fledged response. Also, student answers are not for asking questions, they are meant to contain a response that will help the student asking the question to find out what is wrong with their program. If used to ask for clarification, revise the student answer so that it serves its purpose.

• Finally: DO NOT SHOW OTHER PEOPLE YOUR CODE!!!!! I have seen this in multiple labs. As far as this subject is concerned, know that sharing code, looking at other people's code (even to help them), and letting others see your code is cheating, and will be treated as such without exception. Although debugging a program can be frustrating, cheating will not be tolerated, and we will know if code has been copied.