Rebecca Jackson

# Plagiarism in Programming

Theft of intellectual property is defined by the Federal Bureau of Investigation as "robbing people or companies of their ideas, inventions, and creative expressions" ("Intellectual Property Theft/Piracy," 2016). It 2016 it was considered by the FBI to be a growing threat due to "the rise of digital technologies and Internet file sharing networks" ("Intellectual Property Theft/Piracy"). The most common type of intellectual property theft in an academic setting is plagiarism. The Oregon Institute of Technology defines plagiarism in coding as when you "take program code written by another person and present it as your own work" ("Avoiding Plagiarism," 2018). They also acknowledge that the nature of computer programming it that many ideas within the code will be borrowed from another source. This means that determining what plagiarism is when it comes to computer science is difficult and often unclear.

Since the Oregon Institute of Technology acknowledges that much of programming involves use code which is similar to or copies from another source, they also present guideline for avoiding plagiarism. Different amounts of copied code require different means by which to avoid plagiarism. For example, taking a few lines of code from an example found elsewhere is normally considered fair use and does not require citation, whereas copying a more substantial amount of code requires that its creator be given credit. Since code cannot be quoted like direct quotations from a paper and the creator of the code must be given credit in some instances, copied code is cited within the comments. This makes it clear which sections of code have been borrowed from another source ("Avoiding Plagiarism," 2018). While this is a means of addressing when code is taken from another sources, it does not address all cases which may appear to be plagiarism in coding or their ethical implications.

The main form of protection for software is copyright. Simply placing a copyright sign on your work, however, does not mean that the code is protected. Hypothetically, the copyright sign says that "no one may copy, distribute, display or make adaptations of the work without your permission", however, you may not file a lawsuit against the person using your work without your permission unless the copyright has been registered with the U.S. Copyright Office (Stim, 2014). Even with a software copyright, through a process known as fair use, there are some instances in which the code may be used by others.

Fair use is the idea or belief that the general public may use portions of copyrighted materials to achieve certain goals or tasks. There are four factors used to judge whether or not a use of copyrighted material is what is defined as fair use. These four factors are "the purpose and character of your use, the nature of the copyrighted work, the amount and substantiality of the portion taken, and the effect of the use upon the potential market" (Stim, 2017). Past Supreme Court rulings indicate that an indicator of fair use is the transformation of the copyrighted work into something new. Another factor in determining whether or not your usage of the copyrighted material is considered fair use to consider where the material is coming from. Material taken from published work is more likely to be considered fair use than material taken from unpublished work. The Supreme Court also considers the amount of copied material and how substantial it is in comparison to the rest of the work. Lastly, the Supreme Court looks at the effect of your use on the market of the original work. It is not considered fair use if you take income from the owner of the copyright (Stim, 2017). Anytime the usage of a copyrighted or protected work is considered fair use, there is no legal penalty to the usage of the copied work.

While fair use allows for the legality of many academic uses for copied code, it does not mean that its usage is ethical or allowed based on the standards of the academic institution or professor. For example, in the data mining course at Charleston Southern University, Dr. Babatunde requires the students not "exchange code, tell someone else [their] code, show someone [their] code, ask for code, etc." (2018, p. 5). This is contradictory to the computer science field where programmers frequently share code and are ultimately collaborative (Bidgood, J., & Merrill, J. B., 2017). Further confusion for students is created when they are allowed to discuss a problem but cannot share code. At times this is made even worse when students are told by a professor that "when asking for help, you may have your code viewed by others, but you may not view theirs" (Bidgood, J., & Merrill, J. B., 2017). The grey area in this scenario can be made even worse when students are asking one another for assistance on the same assignment but potentially on different parts of the assignment. In this case, both students end up seeing each other's code even though this was not the original intent.

Ultimately, many students lack the skills needed to troubleshoot code or help another student without first seeing the code. It can be hard to describe where the error is occurring especially when you aren't sure of what is causing it. Students are further hindered by professors

and teachers when they do the same thing. Many professors will simply share their code as a way of telling the student what is wrong with their code. This does not help the student as it simply shows them how to complete the assignment and does not show what they did wrong or how to communicate their inaccuracy and challenges with others. Before students are able to communicate flaws in their code without showing other students the code, they must be taught or shown how to do so.

The New York Times published an article in May of 2017 illustrating the crisis of cheating in the computer science classroom. In one class of 700 at the University of California, Berkeley, the professor found 100 students who had violated his policies on working together or copying code. Brown University had 49 accounts of academic code violation in 2016, more than half of which were from computer science. Stanford University reported in 2015 that twenty percent of students in a single computer science course were flagged for potential cheating. Even the prestigious Harvard University, in their most popular computer science class, had to report more than 60 students to the school for honor code violations (Bidgood, J., & Merrill, J. B., 2017). If such issues exist in these prestigious schools, they are sure to exist elsewhere as well. Until an effective way of teaching students to be better able to communicate their struggles in computer science is found or school continue to crack down on the occurrences, plagiarism will remain an issue in computer science.

Biblically, there is no instance in which plagiarism or theft of intellectual property is considered acceptable. It is logical to assume that no one would want someone to use their code, especially in a way that counteracts the fair use policy put in place by the United States Supreme Court. Using the biblical reference of Luke 6:31 which states that one is to “do to others as you would have them do to you”, we can conclude that any Christian has no means to reason that their theft of another’s code is biblically sound (New International Version). This is a selfish way of thinking of the use of another’s code. To say that you do not copy another’s work merely because you do not wish for someone to copy yours is not a truly Christ like ethical approach. Looking at Mark 10:19, the Jesus says, “You know the commandments: ‘Do not murder, do not commit adultery, do not steal …’” (English Standard Version). Here it is emphasized that any Christian should know that to steal is a sin. Since plagiarism and intellectual property theft are

forms of stealing, this verse shows in a clear and concise manner that both acts are ethically wrong when approaching the issue from a biblical worldview.

While we can determine that plagiarism is wrong and what constitutes plagiarism, it is most important to any student to establish the ways in which one may avoid plagiarism in their computer science courses. The simplest way to ensure that a programming assignment is not plagiarized is to complete the assignment 100 percent on your own. This however is not always possible. In the event that you are unable to determine where the error in the program is occurring, your least likely chance of engaging in plagiarism is to go first to your professor for assistance. In an ideal scenario, the professor is able to help you and give you advice besides to simply use the internet to look up how to solve the error. If you end up having to ask your classmates for assistance, it is best to try and ask specific question which will hopefully prevent you from having to show them your code. Another way to try and avoid plagiarism is that anytime you are using example code from a professor or textbook, be sure to use different variable names and include detailed comments describing what the function of the different parts of the code is. Ultimately, the ease of plagiarism in coding is difficult to combat. Students need careful instruction form professors in order to help them understand what truly constitutes plagiarism in a programming course. In the long run, plagiarism, even if it is not caught, only hurts the students.

Works Cited

Avoiding Plagiarism. (2018). Retrieved from https://www.oit.edu/libraries/help/avoiding-plagiarism

Bidgood, J., & Merrill, J. B. (2017, May 29). As Computer Coding Classes Swell, So Does Cheating. Retrieved from https://www.nytimes.com/2017/05/29/us/computer-science-cheating.html

Charleston Southern University. (2018). *CSCI 442: Data mining [Course syllabus]*: O. Babatunde. Available from Charleston Southern University Blackboard website: csuniv.blackboard.com.

Intellectual Property Theft/Piracy. (2016, November 15). Retrieved from https://www.fbi.gov/investigate/white-collar-crime/piracy-ip-theft

Stim, R. (2014, March 12). Copyrighting Your Software - Why Bother? Retrieved from https://fairuse.stanford.edu/overview/faqs/software/

Stim, R. (2017, March 25). Fair Use. Retrieved from https://fairuse.stanford.edu/overview/fair-use/

Wagner, N. R. (2000). Plagiarism by student programmers. Retrieved from http://www.cs.utsa.edu/~wagner/pubs/plagiarism.html