Coding and Academic Integrity

Guidelines

Here are some key guidelines for university coding assignments:

1. **Do not copy or adapt other people's code, use third-party libraries, or re-use your own pre-existing code without official permission*.**

   If you do, the unit coordinator may determine that you did not complete the assignment properly. The criteria for marking an assignment do not simply assess the combined end product, but your specific personal contribution to it, and you cannot receive marks on the basis of other peoples' work, or your previously-submitted work. Thus, if you use unauthorised material, you could receive far fewer marks than you might otherwise expect, in the normal course of the marking process.

2. **Use in-code comments to give precise references to all external sources.**

   This applies to any external code you copy or adapt, to any third-party libraries you use, to any pre-existing code you wrote yourself, AND to any other reference material on which your code is based, such as algorithm listings, file format descriptions, etc. You must indicate precisely *which* code (from among the code you submit) was obtained from or based on these external sources.

   Missing or inadequate referencing constitutes plagiarism, and you could receive a penalty for academic misconduct.

3. **Do not share code, or work in a group, without official permission*.**

   Working together on the same code, or sharing your code prior to submission, constitutes collusion (except where the assessment explicitly allows or requires groupwork). You could receive a penalty for academic misconduct.

4. **Do not make your assignment work publicly-accessible (until after the semester is over).**

   If you use online services like Github, BitBucket, Google Drive, etc. in preparing your assignment, you must ensure that nobody else can access your work (other than relevant teaching staff and anyone in your group, in the case of legitimate groupwork). Use only "private" repositories.

   If your assignment work is publicly-accessible prior to the due date (or even for several weeks *after* the due date, allowing for possible extensions), this may be considered collusion, and you could receive a penalty for academic misconduct.

* "**Official permission**" means any written communication (e.g., email or online post) from the unit coordinator, or any official unit-related document (e.g., the unit outline or assignment specification), that states explicitly what you are allowed or expected to do.

Principles

There are two principles at stake: **academic integrity**, and **demonstrating learning** (or simply *doing your own work*).

Academic Integrity

You can learn about academic integrity in general at students.curtin.edu.au/essentials/rights/academic-integrity. Essentially, it means not plagiarising, colluding, cheating or falsifying scientific data. To avoid the first three, you must precisely and accurately reference all your sources, if you use any. Coding takes creativity, just like essay and report writing, painting, music composition, etc., and hence if you use other people's code, you must acknowledge *their* creativity.

Academic integrity is about fairness and honesty. When you submit a piece of work (code or otherwise), you're saying that *you* created it. If you submit someone else's work *without saying who actually created it*, you are unfairly and dishonestly claiming the credit for having done that work yourself. Even if you modify someone else's work, it doesn't become your work.

Demonstrating Learning

In an educational setting, it's also essential for you to simply do your own work, and to have others do *their* own work too. This still applies even if you're allowed to work in a group. It may seem like a matter of academic integrity, but it's really about proving that you deserve good marks.

When in the workplace, you won't really have anything to prove in a formal sense. Within the bounds of the law (including contracts and licence conditions), you are free to reuse other people's work as much as you like. You just need to get the job done efficiently and effectively, and code reuse is a perfectly normal way to do that. Nobody need question whether you *could* write the code from scratch if you had to. It's just not necessarily the best use of your time.

At university, you do not have this particular freedom. Assessments (including assignments) are about learning, and proof of learning. They are there to build up your understanding and experience in applying certain concepts, and to allow the university to recognise your achievements through marks and grades. If key parts of your assignment submission are really other people's work, then, even if you reference those people, you still haven't had the proper learning experience, and we can't give you the marks for it.

In some university assignments, you will be given pre-existing code as a starting point, and/or requested to work as part of a team. In such cases, the unit coordinator (UC) has determined that you can achieve the required learning without doing everything yourself. However, this is the prerogative of the UC, and you should not make any assumptions.

How to Reference Code

Here are some approaches to referencing other people's work in a university coding assignment, listed in order of preference. You *must* have official permission from the unit coordinator to use other people's work in the first place, of course.

1. Using Libraries

By their nature, libraries are pre-packaged collections of reusable code (e.g., GTK, Apache Commons, etc.). In a limited sense, they are actually self-referencing, in that in order to use a library you have to specify what it is, or it simply won't work.

Standard libraries do not require referencing (beyond what is required to make them work), as they are the code equivalent of "common knowledge":

```
import java.util.Scanner;
...
```

```
#include <stdio.h>
...
```

Third-party libraries should have more explicit referencing, in a comment at the top of any relevant source code files:

```
// Uses "Apache Commons Lang 3.6", Apache Foundation,
// https://commons.apache.org/proper/commons-lang/

import org.apache.commons.lang3.text;
...
```

```
/* Uses "GTK+ 3.0", The GTK Team, https://www.gtk.org */

#include <gtk/gtk.h>
...
```

2. Copying External Code into Separate Files

If the external code you wish to use has not been pre-packaged into a library, the next best approach is to copy the relevant files or snippets of code whole into a separate directory of your project. If you copy whole files, keep them *unmodified* except to add referencing information to the top as a comment (if not already there).

```
// Obtained from X. Programmer, http://example.com/thecode.java
// (accessed 22 March 2021).

public class ExampleCode
{
    ...
}
```

If you have several code snippets from one or more sources, you might bundle them together and give referencing information for each one:

```
/* This file comprises externally-obtained code. */

#include "example.h"

/* Obtained from X. Programmer,
 * http://stackoverflow.com/questions/12345/abc-xyz
 * (Accessed on 31 May 2021)
 */
int theFunction(double x, double y)
{
    ...
}

/* Obtained from S. Dev,
 * http://github.com/dev/example/tree/master/src/thecode.c
 * (Accessed on 2 June 2021)
 */
void anotherFunction(void)
{
    ...
}
```

3. Copying External Code, or Adapting Algorithms, Into Your Own Files

Sometimes you might need to integrate externally-obtained code into files that also contain your own code. You must clearly show where each externally-obtained snippet starts and stops, and where each one comes from. (A comment at the top is not enough.)

```
/* Parts of this file comprise externally-obtained code. */

public class MyCode
{
    public static void main(String[] args)
    {
        ...

        // Obtained from X. Programmer,
        // http://stackoverflow.com/questions/12345/abc-xyz
        // (Accessed on 31 May 2021).
        for(int i = 0; i < 10; i++)
        {
            doSomething();
        }
        // End of code obtained from X. Programmer

        ...
    }
}
```

If you have not copied anything per se, but rather implemented an algorithm based on an external source, reference this too:

```
/* Algorithm based on Author and Author, "Amazing Algorithms” (2008),
   page 451. */
...
```

4. Constructing your code by modifying an existing template

DO NOT DO THIS for university assessment purposes, UNLESS the lecturer has explicitly provided certain code for this purpose, or explicitly allowed you to find some third-party code. There is no easy way to indicate where your work stops and someone else’s work starts. *If* you do have permission to do this, reference the original source in a comment at the top of the file (if such a comment is not already there):

```
// Based on code from X. Programmer, http://example.com/thecode.java
// (accessed 22 March 2021).

public class ExampleCode
{
    ...
}
```

Plagiarism Detection Myths

There are many myths surrounding detecting, investigation and penalising of academic misconduct, particularly in regards to code. Here are some common ones:

Myth #1. You can cut short the process and just ask to receive a penalty.

A misconduct investigation *must* run to completion, to understand what happened and put it in context. There are (for better or worse) certain minimum timeframes built into it. Moreover, the Inquiry Officer (who will contact you initially) does not have the power to decide on the outcome, only to collect evidence and present it to the Student Discipline Panel (SDP).

Myth #2. The decision is made automatically by software, and I will be penalised if the similarity/originality score is above a certain value.

Misconduct penalties are never applied nor dismissed *automatically*. All decisions are based on human academic judgment, and the final outcome is not based on any auto-generated "similarity/originality score". There may not even *be* a score in the first place.

If investigators have used a tool to help detect plagiarism/collusion, there is no score that you are "allowed to get away with", nor is there a score that guarantees you will be penalised. The score serves only to draw attention to similarities that might need to be investigated. But an investigation can be undertaken anyway, irrespective of the score, if there is one.

Myth #3. I should be able to check the similarity score before submitting. The software should tell me whether I have plagiarised/colluded.

You already know whether you have copied someone else's work. You do not need software to tell you this.

The main use of such a "self-check" feature would be for plagiarisers and colluders to test their techniques for *hiding* what they've done. Someone who has not plagiarised or colluded would never need to use it.

Moreover, a self-check feature doesn't necessarily exist in the software used to help detect plagiarism in *code* assignments. Software certainly cannot tell you whether you have *colluded* or not, prior to submission, as this would require it to already have all the submissions. That's a logical impossibility, unless you already have your classmates' submissions, which in itself is likely to constitute academic misconduct.

Myth #4. I could be unfairly penalised for plagiarism just for following standards, conventions and sample code.

Again, all decisions are based on human academic judgement. Academic staff are well aware of the factors that might legitimately cause two students' code to appear similar. Such factors are always taken into account.

Myth #5. There is only one way to write this code.

There are *thousands* of ways to write it. Coding is creative. There are creative decisions everywhere in software design, in terms of what classes, interfaces, methods and functions you decide to create, what their specific responsibilities of are, how they all communicate, and more. Code would not be copyrightable if there was only one solution to a given problem.

Any two people, working independently, will produce code that is *significantly* different, even if the results of the code are identical.

While there are constraints imposed by the nature of the assignment, there are always many different possible solutions within those constraints. If you and another student have made choices leading to the same code, this is cause for suspicion.

Myth #6. If I change the code, it won't be plagiarism/collusion.

If the code you submit is in any way *based on* someone else's code, you must cite your source. If you do not, you *are* committing plagiarism/collusion, irrespective of any changes you might make.

Myth #7. If I change the code, I won't be *caught* for plagiarism/collusion.

Changing the names of variables, functions, methods, etc. will have little if any effect on code matching tools. Neither will changing formatting, comments, loop types, or any number of other tricks. Indeed, this may be interpreted as a deliberate attempt to avoid being caught, which implies intent to gain unfair advantage.

Myth #8. If I can explain how the code works, I won't be penalised for plagiarism/collusion.

If you have committed plagiarism/collusion, and there is strong evidence for this, your ability to explain the code is largely moot. You may be able to learn how the code works enough to explain it (particularly given the time an investigation often takes), but this is not equivalent to having written the code yourself, and does not make up for the original offence.

Explaining code does not require the same level of comprehension as writing the code in the first place. *Writing* code is the skill that we expect you to develop and demonstrate in a coding assignment.

Some units may require you to explain the code, whether or not there is any (formal) suspicion of misconduct. In such cases, it is vital that you are able to explain it, because an *inability* to explain your own submission is highly suspicious.

Myth #9. I can rebut a plagiarism/collusion allegation by pointing out that individual words and symbols are common or expected.

If an allegation of plagiarism or collusion is put to you, you will likely be shown a comparison of your code against that of someone else. It's important to understand this evidence. What matters is not just the individual matching words and symbols, but the total *length* of matching sections, and uncommonness of *combinations* of matching sections (notable idiosyncracies).

Could two people, working independently, have reproduced those matching sections as a whole? Is there some other reason why they could have occurred legitimately? This is what you must address.

Myth #10. My friends can be trusted with my code. They just want to get some ideas.

Please understand: they are not just “getting ideas”. They are attempting to commit academic misconduct, and so are you if you agree to their request. Whether or not they actually copy your code, you would be helping them gain unfair advantage over other students. A real friend would not put you at risk of a misconduct penalty for their own personal gain.