Plagiarism as an obstacle to academic integrity

A guide for engineering and I.T. students in NUI, Galway

Introduction

Plagiarism is defined by the Academic Council of NUI, Galway as "the act of copying, including or directly quoting from the work of another without adequate acknowledgement" [1]. It has always been an issue in third level education, but the matter is more pressing today, partially because of the ease of access to information on the Internet.

The purpose of this Guide is to inform you of the meaning of plagiarism, strategies to avoid it, and the academic penalties associated with the offence. This document complements the information sessions on plagiarism. You are required to attend the appropriate session. Your lecturers and instructors are your most important source of information and support for all aspects of your study at NUI Galway. Consult them if you are in doubt, and remember that they may have specific requirements for particular modules and assignments in addition to the general guidelines in this document.

The Academic Council has issued a "Code of Practice for Dealing with Plagiarism" [1]. This Guide is based on the Code but is adapted specifically for engineering students. You are required to read the Code itself, which is attached as an Appendix to this guide and is available online [1].

What is plagiarism?

The NUI Galway Code of Practice for Dealing with Plagiarism [1] states:

> "Plagiarism is the act of copying, including or directly quoting from the work of another without adequate acknowledgement, in order to obtain benefit, credit or gain. Plagiarism can apply to many materials, such as words, ideas, images, information, data, approaches or methods. Sources of plagiarism can include books, journals, reports, websites, essay mills, another student, or another person."

Plagiarism may be found to have occurred if a student copies material from a book, journal, website, fellow student or any other source, and then includes this material in a submission for an assignment or project of any kind without giving suitable credit to the original creator of the material. In engineering, the plagiarised material might include text, data, images, ideas or source code. Plagiarism, when found to occur, is subject to the University Code of Conduct.

The following are some possible forms of plagiarism:

- use of material created or provided by another person or agency, such as an "essay mill" or a contract programmer;
- copying the work of another student or individual, with or without that person's consent;
- submission of a student's own work for credit in more than one course;
- copying from a website, book, journal or other document without any citation;
- use of material from a website, book, journal or other document, without an appropriate reference to the exact source.

Some kinds of plagiarism appear to be quite deliberate – for example, the use of "essay mills". However, plagiarism is also committed through misunderstanding. This document and other resources provided for engineering students are intended to help you avoid plagiarism. You should note that plagiarism, whether inadvertent or deliberate, is subject to the University Code of Conduct.

Avoiding plagiarism 1: other students' work

It is common for a group of students to be assigned similar design, analysis or programming tasks, or to record the same data in a laboratory. In these situations, students may be tempted to reduce their workload by submitting work that has been (partly or totally) created by other students. However, the work you submit for course assignments must always be your own, unless you are specifically instructed to work and submit as a group.

Mutual support among students can be a positive part of the learning process. One form of plagiarism, on the other hand, takes place where there is direct collusion or duplication in the submission of assignments, projects etc.

If you have any doubts about whether your work might constitute plagiarism, ask yourself these questions:

- Is your work in your own words, and have you adequately referenced the work of others?
- Are your diagrams and graphs your own, and have you adequately referenced others?
- Are your calculations all your own work?
- You may have discussed the assignment and exchanged ideas with somebody else while working on it. Have you now reached your own complete understanding of it? Could you now carry out the assignment, or a similar one, without help?

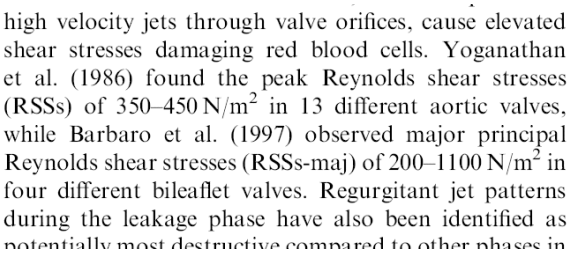
If the answer to any of these questions is not a clear "yes", plagiarism may have occurred.

Note that students who make their work available to be copied (with or without consent) will be dealt with under the same procedures and penalties as students who copy (as per the Code of Practice [1]). You are advised not to provide your written work to other students.

Avoiding plagiarism 2: published literature

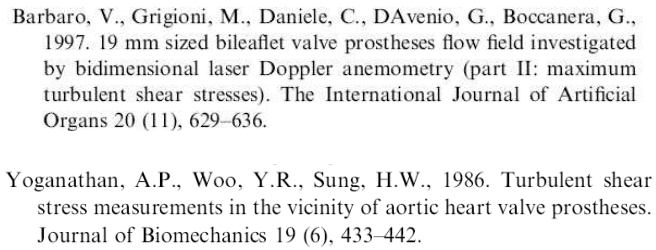
Almost all new engineering work depends in some way on earlier work. It is essential to make use of the work of others in order to make progress. As part of your engineering education, you are expected to develop the skill of finding and using information in the engineering literature (consisting of books, journals, conference proceedings and other publications). This is increasingly important in the later stages of an undergraduate degree, and in postgraduate degrees. However, plagiarism can take place when credit is not given to the creator or author of the original work, implying that credit is due to the author of the current work.

The standard method to acknowledge previous authors in written work consists of two steps. First, a brief *citation* is provided in the text to show where a piece of information or work is due to a previous author. Secondly, *references* are listed (usually at the end of the document) giving full details of the sources used. Figures 1 and 2 (below) show examples of citation and referencing in a paper by Lu *et al*. [3].



high velocity jets through valve orifices, cause elevated shear stresses damaging red blood cells. Yoganathan et al. (1986) found the peak Reynolds shear stresses (RSSs) of 350–450 N/m^2 in 13 different aortic valves, while Barbaro et al. (1997) observed major principal Reynolds shear stresses (RSSs-maj) of 200–1100 N/m^2 in four different bileaflet valves. Regurgitant jet patterns during the leakage phase have also been identified as potentially most destructive compared to other phases in

Figure 1 An extract from the text of a paper by Lu *et al.* [2], including citations to two other publications.



Barbaro, V., Grigioni, M., Daniele, C., DAvenio, G., Boccanera, G., 1997. 19 mm sized bileaflet valve prostheses flow field investigated by bidimensional laser Doppler anemometry (part II: maximum turbulent shear stresses). The International Journal of Artificial Organs 20 (11), 629–636.

Yoganathan, A.P., Woo, Y.R., Sung, H.W., 1986. Turbulent shear stress measurements in the vicinity of aortic heart valve prostheses. Journal of Biomechanics 19 (6), 433–442.

Figure 2 Extracts from the list of references in the paper by Lu *et al.* [2], giving full details of the works cited in the extract shown in Figure 1.

Every citation is associated with a reference. The purposes of citation and referencing are to give credit to authors whose work you have used, to add authority to your own work; and to enable readers to find the background material you have used. References must include full details of the source document – the author(s), year of publication, title of article or book, title, volume and page numbers (for a journal) or publisher (for a book). This guide on plagiarism is not intended to serve as a complete discussion of detailed referencing styles and formats. More guidance on referencing will be given on the courses you take as required.

In engineering, it is rarely necessary to quote the exact words of another author in your work. If you do so, then you must use quotation marks. If you do not use quotation marks, you must *completely* rephrase or summarise the original information in your own words (as well as providing a reference). By doing this, you not only avoid plagiarism, but you show that you have understood the original work, and you can present the information at a level of detail and understanding that are right for your purpose. This is where minor cases of plagiarism often occur, either through mistake or misplaced reliance on the exact words of others where your own are expected..

Engineering often involves searching and utilising existing information sources. However, it is much more than the collection and rearrangement of information – it must involve the creation of new ideas, designs, knowledge or calculations. Obviously, good engineering requires much more than the avoidance of plagiarism.

It's not necessary to provide a reference when you state a piece of "common knowledge" in your work. For example, you might write "Ohm's Law states that $V = IR$" or "Java allows interactive content to be included in web pages". These facts are almost universally known and understood by electronic engineers and information technologists, respectively, and need not be supported by a reference. However, if in doubt, always provide a reference.

Examples

A number of examples that illustrate aspects of plagiarism are presented below. These include examples of work by (hypothetical) students, illustrating different kinds of plagiarism in different kinds of academic work, and showing how good work avoids plagiarism. If you have any questions or doubts about what academic honesty and plagiarism in your particular work, you should ask your lecturer or project supervisor for guidance.

The key to avoiding plagiarism is to distinguish carefully between your own original work and the earlier work of others on which you depend.

We use some formatting conventions in the examples to avoid confusion. Extracts from a published source (which a student might read) are shaded and shown in slightly smaller text, like this:

> this is an extract from a published article, book or website

Extracts from work by a hypothetical student are shown in boxes:

This is a piece of text from a student's report or assignment.

Also, examples of source code will be `shown in Courier font`. Source code will also be shown shaded or in boxes, if it's from a published source or a hypothetical student's work.

Example 1: A design project on electronics cooling

As an example, suppose that a student is designing a cooling system for a particular electronic device. In the early stages of the project, the student quite rightly looks for information in the library and on the internet, and locates a useful article by Chu in a peer-reviewed journal. The article contains the following paragraph supporting the use of "turbulators" on air-cooled electronic modules [3]:

> "As shown in Fig. 2, turbulator strips were placed at appropriate locations on the printed circuit cards to break-up the boundary layer and enhance heat transfer. As shown in Fig. 3, as much as a 23% to 35% reduction in external thermal resistance could be obtained at a fixed flow rate. Of course the added pressure drop due to the turbulators would cause the blowers to shift their operating point and deliver less airflow. Taking this into account, the actual improvement obtained was about 15%. Turbulator strips were used in practice to obtain modest reductions in chip operating temperatures."

Based on this, and other reading and analysis, the student decides to make use of turbulators in his/her design. The student wants to mention Chu's article in his report for several reasons: to support the decision to use turbulators; to show where the idea came from; and to show the examiners that he has done good background reading. The following excerpts from the student's report show various ways in which the student might use the work of Chu.

Good use of a source in example 1

> Chu [3] reviewed methods of cooling electronic modules and found that turbulators can reduce thermal resistance by up to 35% without requiring an increase in air flow (although an increase in pressure drop results). Therefore, turbulators were investigated further as an option for this design.
>
> ⋮
>
> 4 Chu RC, "The Challenges of Electronic Cooling: Past, Current and Future," ASME Journal of Electronic Packaging, 126(4):409-570, 2004

This constitutes a good use of the source article, for several reasons:

- the original author's name is mentioned;
- the number in square brackets after the author's name allows full details of the Chu article to be found in a list of references at the back of the document;
- the student has taken only the essential elements of Chu's paragraph, and those which are relevant to the current project;
- the student appears to understand the source article and its importance to his/her project;
- the student has presented the important information concisely in her/his own words (this is called paraphrasing).

Each one of these elements is essential.

Clear Plagiarism (1) in example 1

> External resistance can be reduced by employing boundary layer turbulators. Turbulator strips are placed at appropriate locations on the printed circuit cards to break-up the boundary layer and enhance heat transfer. As much as a 23% to 35% reduction in external thermal resistance can be obtained at a fixed flow rate. Of course the added pressure drop due to the turbulators would cause the blowers to shift their operating point and deliver less airflow. Taking this into account, the actual improvement obtained is about 15%. Turbulator strips are used in practice to obtain modest reductions in chip operating temperatures.

In this example, the student has copied a considerable amount of text from the source with just a few word changes. He/she has made no attempt to give credit to the original author, or to make the content suit the current work.

Clear Plagiarism (2) in example 1

> Chu [3] has shown that external resistance can be reduced by employing boundary layer turbulators. Turbulator strips are placed at appropriate locations on the printed circuit cards to break-up the boundary layer and enhance heat transfer. As much as a 23% to 35% reduction in external thermal resistance can be obtained at a fixed flow rate. Of course the added pressure drop due to the turbulators would cause the blowers to shift their operating point and deliver less airflow. Taking this into account, the actual improvement obtained is about 15%. Turbulator strips are used in practice to obtain modest reductions in chip operating temperatures.
>
> ⋮
>
> 4 Chu RC, “The Challenges of Electronic Cooling: Past, Current and Future,” ASME Journal of Electronic Packaging, 126(4):409-570, 2004

This is the same as the previous example, but the student has prefaced the extract by citing its source. However, the student has made not made it clear that the words (as well as the ideas) are due to Chu. Furthermore, as the student has not distilled the essential information from the original, it may be the case that she/he does not in fact understand it.

Direct quotation in example 1 – plagiarism-free but lazy

> Chu [3] states:
>
> “In instances where this was not possible the external resistance was reduced by employing boundary layer turbulators. As shown in Fig. 2, turbulator strips were placed at appropriate locations on the printed circuit cards to break-up the boundary layer and enhance heat transfer. As shown in Fig. 3, as much as a 23% to 35% reduction in external thermal resistance could be obtained at a fixed flow rate. Of course the added pressure drop due to the turbulators would cause the blowers to shift their operating point and deliver less airflow. Taking this into account, the actual improvement obtained was about 15%. Turbulator strips were used in practice to obtain modest reductions in chip operating temperatures.”
>
> ⋮
>
> 4 Chu RC, “The Challenges of Electronic Cooling: Past, Current and Future,” ASME Journal of Electronic Packaging, 126(4):409-570, 2004

This student has placed quotation marks around Chu's words, and given a proper reference to the source of the text. Strictly speaking, this is not plagiarism. However, it is inappropriate work – again it suggests that the student has not understood the original paper properly. Avoiding plagiarism is not enough to ensure high-quality work. Direct quotation should only be used if the wording of the original is special and unique in some way – and this is rarely, if ever, the case in engineering, where the ideas are almost always more important than the words. In this case, the student could not be penalised for plagiarism, but would probably not earn high marks for the work.

Example 2: A literature survey on clustering algorithms

As another example, suppose that as part of a literature survey a student is investigating algorithms (the logical methods behind programs) that might be useful for analysing complex bioinformatics data sets. In a book by Raychaudhuri [4], the student has found the following explanation of k-means clustering:

> "One of the simplest clustering methods is k-means clustering. It is very easy to implement. K-means clustering requires a parameter k, the number of expected clusters. Correct selection of k can dramatically affect the final clustering results and unfortunately it is often difficult to know a priori what an appropriate choice for k is.
>
> Initially k cluster centres, c1,….ck, are randomly selected expression profiles taken from the data set. In each iteration of the algorithm, the distances between each of the genes and the k centres are calculated using the pre-selected distance metric; genes are then assigned to the cluster whose centre they are nearest to.
>
> For each gene xj and each centre ci:
>
> dj,i = D(xj, ci)
>
> cluster(xj) = arg mini(dj, i)
>
> After the genes have been assigned to clusters, the cluster centres are recomputed by taking the average of the genes assigned to the cluster. In the subsequent iteration, genes are again assigned to the cluster whose centre they are nearest to and then the centres are recalculated; this process is repeated until the algorithm converges. Unfortunately the algorithm converges to a local minimum, and is very sensitive to the initial random selection of starting centres."

The student decides to make use of this explanation as part of her/his report on algorithms for analysing complex data sets.

Good use of a source in example 2

According to Raychaudhuri [5], k-means clustering starts by randomly selecting k cluster centres. Several iterations then follow until eventually the algorithm converges. In each iteration of the algorithm, the distance between any one gene and all of the cluster centres is computed using a pre-defined distance metric. Each gene is then assigned to the cluster that minimises this distance. Each cluster centre is then recomputed by averaging over all the genes now assigned to that cluster. The algorithm converges to a local rather than a global minimum (which may in fact not be optimal), and is quite sensitive to the initial selection of centres. Further, the number of expected centres, k, may be difficult to determine in advance.

⋮

5 Raychaudhuri, S. "Computational Text Analysis for functional genomics and bioinformatics", Oxford University Press, 2006

The student has provided both the author's name and a citation (i.e. the number in square brackets) to a location further on in the student's document where the full reference may be found. The student has also made an attempt to summarise the given explanation concisely, entirely in her own words, outlining the operation of the algorithm and any significant caveats. In the context of a literature survey, this kind of summarisation and criticism is what is being sought by readers or examiners.

Clear plagiarism (1) in example 2

> K-means clustering requires a parameter k, the number of expected clusters. Correct selection of k can dramatically affect the final clustering results and unfortunately it is often difficult to know what an appropriate choice for k is.
>
> **Randomly selected expression profiles taken from the data set are initially k cluster centres, c1,….ck.** In each iteration of the algorithm, the distances between each of the genes and the k centres are calculated using the pre-selected distance metric; genes are then assigned to the cluster whose centre they are nearest to.
>
> For each gene xj and each centre ci:
>
> dj,i = D(xj, ci)
>
> cluster(xj) = arg mini(dj, i)
>
> After the genes **were** assigned to clusters, the cluster centres **were** recomputed by taking the average of the genes assigned to the cluster. In the subsequent iteration, genes **were** again assigned to the cluster whose centre they **were** nearest to and then the centres **were** recalculated; this process **was** repeated until the algorithm **converged**. Unfortunately the algorithm converges to a local minimum, and is very sensitive to the initial random selection of starting centres.
>
> ⋮
>
> 5 Raychaudhuri, S. "Computational Text Analysis for functional genomics and bioinformatics", Oxford University Press, 2006

The above material is directly copied without citing the source. The only changes evident are (1) use of the past tense in the last paragraph (bold text) (2) dropping the first two sentences and (3) swapping two portions of exactly one sentence (bold text). All the rest is identical with the original material. Furthermore, although a reference is provided at the end of the document, there is no citation in the text to link this part of the student's report specifically to the original work by Raychaudhuri.

Clear plagiarism (2) in example 2

According to Raychaudhuri [5], k-means clustering is very easy to implement.

K-means clustering requires a parameter k, the number of expected clusters. Correct selection of k can dramatically affect the final clustering results and unfortunately it is often difficult to know what an appropriate choice for k is.

Randomly selected expression profiles taken from the data set are initially k cluster centres, c1,….ck. In each iteration of the algorithm, the distances between each of the genes and the k centres are calculated using the pre-selected distance metric; genes are then assigned to the cluster whose centre they are nearest to.

For each gene xj and each centre ci:

dj,i = D(xj, ci)

cluster(xj) = arg mini(dj, i)

After the genes **were** assigned to clusters, the cluster centres **were** recomputed by taking the average of the genes assigned to the cluster. In the subsequent iteration, genes **were** again assigned to the cluster whose centre they **were** nearest to and then the centres **were** recalculated; this process **was** repeated until the algorithm **converged**. Unfortunately the algorithm converges to a local minimum, and is very sensitive to the initial random selection of starting centres.

⋮

5 Raychaudhuri, S. “Computational Text Analysis for functional genomics and bioinformatics”, Oxford University Press, 2006

All that differs from the previous example is the source citation. The reach of that citation is unclear, so that it is impossible for a reader to know how much of the student’s writing is based on Raychaudhuri’s paper. The words remain very close to the original.

Direct quotation in example 2 – plagiarism-free but lazy

Raychaudhuri [5] states:

"One of the simplest clustering methods is k-means clustering. It is very easy to implement.

K-means clustering requires a parameter k, the number of expected clusters. Correct selection of k can dramatically affect the final clustering results and unfortunately it is often difficult to know a priori what an appropriate choice for k is.

Initially k cluster centres, c1,….ck, are randomly selected expression profiles taken from the data set. In each iteration of the algorithm, the distances between each of the genes and the k centres are calculated using the pre-selected distance metric; genes are then assigned to the cluster whose centre they are nearest to.

For each gene xj and each centre ci:

dj,i = D(xj, ci)

cluster(xj) = arg mini(dj, i)

After the genes have been assigned to clusters, the cluster centres are recomputed by taking the average of the genes assigned to the cluster. In the subsequent iteration, genes are again assigned to the cluster whose centre they are nearest to and then the centres are recalculated; this process is repeated until the algorithm converges. Unfortunately the algorithm converges to a local minimum, and is very sensitive to the initial random selection of starting centres."

⋮

5 Raychaudhuri, "Computational Text Analysis for functional Genomics and Bioinformatics", OUP, 2006

This approach is not plagiarism in the formal sense. There are quotation marks and a reference . However, this approach should be used sparingly. The student has made no attempt to formulate the ideas in the student's own words – and it is the transmission of ideas through the student's own words that is generally of interest in a report. A direct quotation should be reserved for occasions where the actual direct wording of the source is immediately relevant.

Example 3: Programming a Bubble Sort Algorithm

In the case of code, a distinction must be drawn between the algorithm (the logical method behind a program) and the code that implements the algorithm (the actual lines of C++, Fortran etc.). An algorithm may be very well known and presented in a standard way, for example bubble sort, and the code that implements it may not vary significantly between different programmers except perhaps in the use of things like function and variable names.

“Bubble sort” is a very well-known algorithm for sorting data into a required order The following code, drawn from [6], shows elements of a current web-based example of bubble sort code to sort the elements of the array a in order of increasing numerical value:

```
for (i=0; i<n-1; i++) {
  for (j=0; j<n-1-i; j++)
    if (a[j+1] < a[j]) {
      tmp = a[j];
      a[j] = a[j+1];
      a[j+1] = tmp;
  }
}
```

The following code, drawn from [7], shows elements of an earlier C implementation of bubble sort:

```
for (i = 0 ; i < n-1 ; ++i)
  for (k = 0 ; k < n -(i + 1); ++k)
     if (elem[k] > elem[k+1])
     {
      help = elem[k];
      elem[k] = elem[k+1];
      elem[k+1] = help;
  }
```

Both examples carry out the same work, but there are differences:

(1) Variable names: `j`, `a`, `tmp` `versus` `k`, `elem`, `help`

(2) `i++` versus `++i`

(3) `n-1-i` versus `n-(i +1)`

(4) `a[j+1] < a[j]` versus `elem[k] > elem[k+1]`

(5) use of brackets

(6) Indentation

So even in a case where essentially the same work is being done by two individuals, and the work involves a small programming task, differences will always be expected in their code. Different individuals, working independently, will always come up with slightly different solutions to the same problem if they. In the example above, there is no question of plagiarism.

Example 4: Programming a game

When we consider a more complex project aimed at developing a tic-tac-toe (X and O) playing program, we would expect a lot of differences between individual approaches. Consider chapter 13 of the text on game code development written by Jacob Habgood and Mark Overmars [8].

In this chapter, the authors describe how to construct a computer program for playing a game of tic-tac-toe. Tic-tac-toe is a simple turn-by-turn game played in a three-by-three grid where the aim is to get three identical symbols in a row, column, or diagonal. It is often played with the symbols X and O.

A number of functions that implement the game are discussed in the chapter. The functions are written in a scripting language associated with the Game Maker environment. A zero in a board position indicates that that position is empty, a one that the player is occupying it and a two that the computer is occupying it.

The following function `scr_find_move()`adjusts the computer's play by considering the relative number of wins. If the player is playing poorly, a random move is made. If the player is improving, three tactics are gradually introduced: looking for a winning move, blocking a winning move on the part of the player, and attempting to occupy the centre.

```
scr_find_move()
{
  var level;
  level = (score_player+1) / (score_computer+1);
  // find the best move
  if (level > 0.5)
    { if scr_find_win() exit; }
  if (level > 0.8)
    { if scr_find_lose() exit; }
  if (level > 1.2)
    { if scr_find_center() exit; }
  scr_find_random();
}
```

The following function `scr_find_win()`checks all board positions for the next move, and, if a position is free, provisionally moves there to see if moving to that position will result in a win for the computer. It makes use of another function that performs the actual check. If a win does not result, the move is unmade.

```
scr_find_win()
{
  var i,j;
  // see whether there is a winning move
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 2;
        if scr_check_computer_win() return true;
        field[i,j] = 0;
      }
  return false;
}
```

The following function `scr_check_computer_win()`checks to see if there is a win for the computer by looking for three '2' symbols in a row in any of a row, column, or diagonal.

```
scr_check_computer_win()
{
  // check wether there is a row of 2's
  if (field[0,0]==2 && field[0,1]==2 && field[0,2]==2) return true;
  if (field[1,0]==2 && field[1,1]==2 && field[1,2]==2) return true;
  if (field[2,0]==2 && field[2,1]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,0]==2 && field[2,0]==2) return true;
  if (field[0,1]==2 && field[1,1]==2 && field[2,1]==2) return true;
  if (field[0,2]==2 && field[1,2]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,1]==2 && field[2,2]==2) return true;
  if (field[0,2]==2 && field[1,1]==2 && field[2,0]==2) return true;
  return false;
}
```

The following function `scr_find_lose()`checks to see if the player can win on the next move. It looks at all board positions and, if a position is free, simulates the player moving there to see if moving to that position will result in a win for the player. If a win would result, the program itself occupies the position with a '2' symbol to block the win. It makes use of another function to check the win condition. If a win would not result for the player, the simulated move is unmade.

```
scr_find_lose()
{
  var i,j;
  // see whether there is a winning move for the player
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 1;
        if scr_check _player_win()
          { field[i,j] = 2; return true; }
        field[i,j] = 0;
      }
  return false;
}
```

The following function `scr_check_player_win()`checks to see if a win for the player results. It does this by looking for three ‘1’ symbols in a row in any of a row, column, or diagonal.

```
scr_check_player_win()
{
  // check wether there is a row of 1's
  if (field[0,0]==1 && field[0,1]==1 && field[0,2]==1) return true;
  if (field[1,0]==1 && field[1,1]==1 && field[1,2]==1) return true;
  if (field[2,0]==1 && field[2,1]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,0]==1 && field[2,0]==1) return true;
  if (field[0,1]==1 && field[1,1]==1 && field[2,1]==1) return true;
  if (field[0,2]==1 && field[1,2]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,1]==1 && field[2,2]==1) return true;
  if (field[0,2]==1 && field[1,1]==1 && field[2,0]==1) return true;
  return false;
}
```

The following function `scr_find_center()`looks fifty per cent of the time to see if the centre is free and, if it is, it occupies the centre with a ‘2’ symbol.

```
scr_find_center()
{
  // half the time try to play the center position
  if (random(2) < 1 && field[1,1] == 0)
    { field[1,1] = 2; return true; }
  return false;
}
```

The following function `scr_find_random()`finds a random position on the board that is free and occupies it with a ‘2’ symbol.

```
scr_find_random()
{
  var i,j;
  // find a random empty position
  while(true)
  {
    i = floor(random(3));
    j = floor(random(3));
    if (field[i,j] == 0)
    {
      field[i,j] = 2;
      exit;
    }
  }
}
```

Now consider a student asked to implement Tic-Tac-Toe for a programming project. She/he might read this chapter and then go on to develop her/his own version of the game. Quite apart from syntactic issues, she/he would have plenty of opportunities to take alternative approaches in her version of the game, for example:

- Develop an entirely different way of approaching the whole game , e.g. minimax
- Develop different and possibly better versions of several of the above functions
- Develop a new function for a corner move

Etc. etc.

The point is that she/he should treat the example code as informing her/his own work, not simply relying directly on the code itself

Clear plagiarism (1) in example 4

In a student's code for a tic-tac-toe project, the following is found:

```
scr_find_move()
{
  var level;
  level = (score_player+1) / (score_computer+1);
  // find the best move
  if (level > 0.5)
    { if scr_find_win() exit; }
  if (level > 0.8)
    { if scr_find_lose() exit; }
  if (level > 1.2)
    { if scr_find_center() exit; }
  scr_find_random();
}


scr_find_win()
{
  var i,j;
  // see whether there is a winning move
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 2;
        if scr_check_computer_win() return true;
        field[i,j] = 0;
      }
  return false;
}


scr_check_computer_win()
{
  // check wether there is a row of 2's
  if (field[0,0]==2 && field[0,1]==2 && field[0,2]==2) return true;
  if (field[1,0]==2 && field[1,1]==2 && field[1,2]==2) return true;
  if (field[2,0]==2 && field[2,1]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,0]==2 && field[2,0]==2) return true;
  if (field[0,1]==2 && field[1,1]==2 && field[2,1]==2) return true;
  if (field[0,2]==2 && field[1,2]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,1]==2 && field[2,2]==2) return true;
  if (field[0,2]==2 && field[1,1]==2 && field[2,0]==2) return true;
  return false;
}


scr_find_lose()
{
  var i,j;
  // see whether there is a winning move for the player
```

```
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 1;
        if scr_check _player_win()
          { field[i,j] = 2; return true; }
        field[i,j] = 0;
      }
  return false;
}


scr_check_player_win()
{
  // check wether there is a row of 1's
  if (field[0,0]==1 && field[0,1]==1 && field[0,2]==1) return true;
  if (field[1,0]==1 && field[1,1]==1 && field[1,2]==1) return true;
  if (field[2,0]==1 && field[2,1]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,0]==1 && field[2,0]==1) return true;
  if (field[0,1]==1 && field[1,1]==1 && field[2,1]==1) return true;
  if (field[0,2]==1 && field[1,2]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,1]==1 && field[2,2]==1) return true;
  if (field[0,2]==1 && field[1,1]==1 && field[2,0]==1) return true;
  return false;
}

scr_find_center()
{
  // half the time try to play the center position
  if (random(2) < 1 && field[1,1] == 0)
    { field[1,1] = 2; return true; }
  return false;
}


scr_find_random()
{
  var i,j;
  // find a random empty position
  while(true)
  {
    i = floor(random(3));
    j = floor(random(3));
    if (field[i,j] == 0)
    {
      field[i,j] = 2;
      exit;
    }
  }
}
```

Notice that in this case all of the functions from chapter 13 of [8] have been directly re-used. No attempt has been made to develop any other functions; rather the original material is simply re-used directly and completely. The functions are presented in the same order, with all the same names, punctuation and indentation, and contain exactly the same control constructs and comments. The same symbols have been used, namely ‘0’, ‘1’ and ‘2’ to designate occupying board positions, and exactly the same cutoffs have been used in the first function, namely 0.5, 0.8 and 1.2. The exact same calls are made to helper functions as in the original.

Even in the case of bubble sort, where a standard approach exists, we saw that variation is to be expected in the use of variable names, ways of comparing things, and things like brackets and indentation. In the case of this tic-tac-toe playing program, not even this level of syntactic variation is present in any one function. In addition, the seven functions have been directly re-used.

Clear plagiarism (2) in example 3

In the code associated with another student project, the following is found:

```
//based on example by Habgood and Overmars[8]

scr_find_move()
{
  var level;
  level = (score_player+1) / (score_computer+1);
  // find the best move
  if (level > 0.5)
    { if scr_find_win() exit; }
  if (level > 0.8)
    { if scr_find_lose() exit; }
  if (level > 1.2)
    { if scr_find_center() exit; }
  scr_find_random();
}



scr_find_win()
{
  var i,j;
  // see whether there is a winning move
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 2;
        if scr_check_computer_win() return true;
        field[i,j] = 0;
      }
  return false;
}


scr_check_computer_win()
{
  // check wether there is a row of 2's
  if (field[0,0]==2 && field[0,1]==2 && field[0,2]==2) return true;
  if (field[1,0]==2 && field[1,1]==2 && field[1,2]==2) return true;
  if (field[2,0]==2 && field[2,1]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,0]==2 && field[2,0]==2) return true;
  if (field[0,1]==2 && field[1,1]==2 && field[2,1]==2) return true;
  if (field[0,2]==2 && field[1,2]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,1]==2 && field[2,2]==2) return true;
  if (field[0,2]==2 && field[1,1]==2 && field[2,0]==2) return true;
  return false;
}

scr_find_lose()
{
  var i,j;
  // see whether there is a winning move for the player
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
```

```
      if (field[i,j] == 0)
      {
        field[i,j] = 1;
        if scr_check _player_win()
          { field[i,j] = 2; return true; }
        field[i,j] = 0;
      }
  return false;
}

scr_check_player_win()
{
  // check wether there is a row of 1's
  if (field[0,0]==1 && field[0,1]==1 && field[0,2]==1) return true;
  if (field[1,0]==1 && field[1,1]==1 && field[1,2]==1) return true;
  if (field[2,0]==1 && field[2,1]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,0]==1 && field[2,0]==1) return true;
  if (field[0,1]==1 && field[1,1]==1 && field[2,1]==1) return true;
  if (field[0,2]==1 && field[1,2]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,1]==1 && field[2,2]==1) return true;
  if (field[0,2]==1 && field[1,1]==1 && field[2,0]==1) return true;
  return false;
}




scr_find_center()
{
  // half the time try to play the center position
  if (random(2) < 1 && field[1,1] == 0)
    { field[1,1] = 2; return true; }
  return false;
}


scr_find_random()
{
  var i,j;
  // find a random empty position
  while(true)
  {
    i = floor(random(3));
    j = floor(random(3));
    if (field[i,j] == 0)
    {
      field[i,j] = 2;
      exit;
    }
  }
}
```

All that differs form the previous example is the initial citation to Habgood and Overmars. It is claimed that the code is ‘based’ on the work in [8], but the reach of the citation is unclear. The student has not demonstrated her/his understanding of the material. The code remains exactly as in the first example of plagiarism in example three, and exactly as it appeared in the original book chapter [8].

Direct use of cited code – plagiarism-free but lazy

In the code associated with another project, the following is found:

```
//all code is from an example game by Habgood and Overmars[8]
// Habgood J, Overmars M, "The Game Maker's Apprentice: Game
// Development for Beginners", APress, Berkeley, 2006
scr_find_move()
{
  var level;
  level = (score_player+1) / (score_computer+1);
  // find the best move
  if (level > 0.5)
    { if scr_find_win() exit; }
  if (level > 0.8)
    { if scr_find_lose() exit; }
  if (level > 1.2)
    { if scr_find_center() exit; }
  scr_find_random();
}

scr_find_win()
{
  var i,j;
  // see whether there is a winning move
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 2;
        if scr_check_computer_win() return true;
        field[i,j] = 0;
      }
  return false;
}


scr_check_computer_win()
{
  // check wether there is a row of 2's
  if (field[0,0]==2 && field[0,1]==2 && field[0,2]==2) return true;
  if (field[1,0]==2 && field[1,1]==2 && field[1,2]==2) return true;
  if (field[2,0]==2 && field[2,1]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,0]==2 && field[2,0]==2) return true;
  if (field[0,1]==2 && field[1,1]==2 && field[2,1]==2) return true;
  if (field[0,2]==2 && field[1,2]==2 && field[2,2]==2) return true;
  if (field[0,0]==2 && field[1,1]==2 && field[2,2]==2) return true;
  if (field[0,2]==2 && field[1,1]==2 && field[2,0]==2) return true;
  return false;
}

scr_find_lose()
{
  var i,j;
  // see whether there is a winning move for the player
  for (i=0; i<=2; i+=1)
    for (j=0; j<=2; j+=1)
      if (field[i,j] == 0)
      {
        field[i,j] = 1;
        if scr_check _player_win()
          { field[i,j] = 2; return true; }
        field[i,j] = 0;
```

```
      }
  return false;
}

scr_check_player_win()
{
  // check wether there is a row of 1's
  if (field[0,0]==1 && field[0,1]==1 && field[0,2]==1) return true;
  if (field[1,0]==1 && field[1,1]==1 && field[1,2]==1) return true;
  if (field[2,0]==1 && field[2,1]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,0]==1 && field[2,0]==1) return true;
  if (field[0,1]==1 && field[1,1]==1 && field[2,1]==1) return true;
  if (field[0,2]==1 && field[1,2]==1 && field[2,2]==1) return true;
  if (field[0,0]==1 && field[1,1]==1 && field[2,2]==1) return true;
  if (field[0,2]==1 && field[1,1]==1 && field[2,0]==1) return true;
  return false;
}


scr_find_center()
{
  // half the time try to play the center position
  if (random(2) < 1 && field[1,1] == 0)
    { field[1,1] = 2; return true; }
  return false;
}


scr_find_random()
{
  var i,j;
  // find a random empty position
  while(true)
  {
    i = floor(random(3));
    j = floor(random(3));
    if (field[i,j] == 0)
    {
      field[i,j] = 2;
      exit;
    }
  }
}
```

In this case, the citation makes it clear that all the code is being re-used. Strictly speaking, this is not plagiarism. However, even in cases where authors explicitly permit the re-use of their material, (see [9] for a discussion of such permissions) it is most unlikely that in the context of an assignment or project a supervisor expected or sanctioned such complete and total re-use of example code. Code in a project, like words in a report, is understood to represent the student's own work and understanding. The direct use of cited code, like directly quoting text from a source document, does not in any way demonstrate the student's own contribution. If in any doubt whatsoever, consult with your project supervisor or relevant lecturer.

Sources of further information

As part of your engineering course, you receive further guidance on technical writing, which will complement several of the issues raised here. Many books and web-based resources are also available. In particular, the Internet Detective [10] and the Acadia University Tutorial on Plagiarism [11] provide excellent interactive tutorials.

Procedures and Penalties for Dealing with Plagiarism

Procedures and penalties with plagiarism are outlined in the University Code of Practice [1], which you are required to read. The plagiarism advisors for the Faculty of Engineering are Dr. Nathan Quinlan (Mechanical Engineering) and Dr. Conn Mulvihill (Information Technology). This section contains a summary of the main points of the Code, with details of particular procedures in the College of Engineering and Informatics.

A lecturer who suspects plagiarism reports it to a plagiarism advisor. The advisor follows the procedure described in the Code of Practice to determine if there is a case to be made, and the severity of the case. If there is a case to me made, the advisor will meet the student, and decide on the appropriate penalty (if any). Penalties range from a formal warning to referral to the University Discipline Committee. Penalties are more severe on second and subsequent incidents of plagiarism.

For full details, please refer to the Code of Practice.

The MSc in Software and Information Systems

The MScSIS is a distance learning course provided jointly by NUI, Galway and Regis University, USA. The case will be processed by the university (NUI, Galway or Regis) responsible for the module in which the suspected plagiarised work is submitted, according to standard practice in that university. Because of the nature of the programme, the following slightly modified procedure will be followed.

1. If a module facilitator decides to formally report a suspected plagiarism case, s(he) will send to the faculty plagiarism advisor a copy of the submitted work, together with any evidence. This might include a copy of the original text, data, source code, images (or otherwise) along with a reference to the original source (e.g. text book, web site).

2. The facilitator will send an e-mail to the student (with a CC: to the plagiarism advisor) informing him/her that plagiarism is suspected on the assignment and that a plagiarism advisor will send the student an e-mail to arrange an interview, according to the NUI, Galway Code of Practice for Dealing with Plagiarism. The facilitator will advise the student that the assignment will not be marked until the case is investigated.

3. The initial communication from the plagiarism advisor to the student will be by e-mail, to serve as the written notification required by the code, and to suggest possible times for an interview, which will take place by telephone conference. The e-mail will be copied to the module facilitator who raises the suspicion of plagiarism, and to the MScSIS academic coordinator.

4. The student will be required to reply to the initial e-mail within 48 hours. In the student's reply, (s)he will provide the times and phone number at which (s)he will be available for interview, and the name of the "friend" (if any) who will accompany him or her in the interview (as allowed by the Code of Practice). (S)he will also state the relationship of that person to the student (e.g. fellow student, legal representative).

5. The staff member accompanying the plagiarism advisor in the interview will be the module facilitator or a member of the MScSIS programme academic board.

6. The MScSIS academic coordinator and the module facilitator will be informed of the outcome.

7. 7In the normal course of events, the entire process would be expected to be completed within twenty working days of receipt of the completed report form by the plagiarism advisor.

References

1 –, “Code of Practice for dealing with plagiarism,” National University of Ireland Galway, 2012, http://www.nuigalway.ie/plagiarism/ [accessed 25-2-2013]

2 Lu PC, Lai HC, Liu JS, “A reevaluation and discussion on the threshold limit for hemolysis in a turbulent shear flow”, Journal of Biomechanics 34:1361–1364, 2001

3 Chu RC, “The Challenges of Electronic Cooling: Past, Current and Future,” ASME Journal of Electronic Packaging, 126(4):409-570, 2004

4 Raychaudhuri, S “Computational Text Analysis for functional genomics and bioinformatics”, Oxford University Press, 2006

5 Hausner A, “Bubble Sort”, http://www.cs.princeton.edu/~ah/alg_anim/gawain-4.0/BubbleSort.html, [accessed 12-02-2008]

6 Schroder K, “C”, Addison Wesley, 2002

7 Habgood J, Overmars M, “The Game Maker’s Apprentice: Game Development for Beginners”, APress, Berkeley, 2006.

8 http://creativecommons.org [accessed 12-02-2008]

9 Place, E., Kendall, M., Hiom, D., Booth, H., Ayres, P., Manuel, A., Smith, P. (2006) "Internet Detective: Wise up to the Web", 3rd edition, Intute Virtual Training Suite, [online]. Available from: http://www.vts.intute.ac.uk/detective/ [Accessed 12-02-2008]

10 Tutorial on Plagiarism, Vaughan Memorial Library, Acadia University, Nova Scotia, Canada, http://library.acadiau.ca/tutorials/plagiarism/ [Accessed 12-02-2008]

Appendix

The following is the text of the official NUI, Galway Code of Practice for Dealing with Plagiarism [1].

Code of Practice for Dealing with Plagiarism

This Plagiarism Code comes into effect at the start of the 2012/13 academic year.

1.0 Purpose

To set out the code of practice for dealing with issues of student plagiarism.

2.0 Description

Plagiarism is the act of copying, including or directly quoting from the work of another without adequate acknowledgement, in order to obtain benefit, credit or gain. Plagiarism can apply to many materials, such as words, ideas, images, information, data, approaches or methods. Sources of plagiarism can include books, journals, reports, websites, essay mills, another student, or another person.

Self-plagiarism, or auto-plagiarism, is where a student re-uses work previously submitted to another course within the University or in another Institution.

All work submitted by students for assessment, for publication or for (public) presentation, is accepted on the understanding that it is their own work and contains their own original contribution, except where explicitly referenced using the accepted norms and formats of the appropriate academic discipline.

Plagiarism can arise through poor academic practice or ignorance of accepted norms of the academic discipline. Schools should ensure that resources and education around good academic practice is available to students at all levels.

The Plagiarism Penalty Grid (included in this document) will be made available to all students.

Cases in which students facilitate others to copy their work shall also be subject to the procedures outlined here.

2.1 Procedures

Each School will appoint at least one plagiarism advisor, who is normally a member of academic staff. These advisors are Designated Authorities, as described in the Student Code of Conduct, and have responsibility and authority for dealing with suspected and reported cases of plagiarism.

A list of the current plagiarism advisors will be maintained and made available to all academic staff of the University.

A member of teaching staff who suspects plagiarism is welcome to speak with an appropriate plagiarism advisor, in confidence, about the case. At this point, the staff

member is free not to continue with a formal report.

If a staff member decides to formally report a suspected case of plagiarism, a short report shall be prepared including a (marked-up) copy of the student work, along with any evidence for suspecting plagiarism. This report should be forwarded to the plagiarism advisor.

The plagiarism advisor shall conduct an initial investigation of the alleged plagiarism, to determine if there is a case to be made. If the advisor concludes that there is no case of plagiarism, the reporting member of staff will be notified, with a clear statement of the reasons for the decision.
If the plagiarism advisor decides that the case is one of plagiarism, he/she will make an initial assessment of the case using the penalty grid (step 1).

If the points, according to the penalty grid, are in the lower two bands (up to 379) the advisor may conduct an informal interview with the student to discuss the suspected case. If the advisor is satisfied that the case exists, an appropriate penalty will be selected from the grid (step 2).

If the points, according to the penalty grid, are more than 524, the advisor should refer the case to the discipline committee, in accordance with the Student Code of Conduct.

In all other cases (points in the bands 380-524), the student will be invited to attend an interview with the plagiarism adviser and an additional member of staff. The invitation may be by email or letter, and will include an explanation of the purpose of the meeting, including a copy of the marked-up piece of work. The student may be accompanied at the interview by a “friend”.

The additional member of staff may be another plagiarism advisor, the member of staff who reported the case, or another senior member of staff from the School.

Where a student does not engage with the process, by not responding or by refusing to attend an interview, the case will be referred to the discipline committee.

At the interview, the student will be given a clear explanation of what has been alleged, shown a copy of his/her work, given the opportunity to justify the work and be invited to admit or deny responsibility.

Following the interview, if the advisor is satisfied that the case exists, an appropriate penalty will be selected from the grid (step 2). After a penalty has been decided, the advisor will perform a fairness check to consider the impact of the penalty on the student’s overall performance. If the impact is incommensurate with the offence, the advisor may choose to adjust the penalty. In all cases, the student will be notified by the advisor, in writing, of the decision and any penalty imposed.

The plagiarism advisor will write a report, recording the decision and any penalty, which should be lodged centrally. This report is confidential and will not reflect upon the student’s record. It will be used to determine if a second or subsequent offence has occurred, and for statistical information only.

It may be appropriate for incidents of plagiarism to be made known to relevant academic and support staff where this is required for the proper administration of academic

programmes and academic decision making. Such sharing of information with appropriate staff does not breach confidentiality.

3.0 Related Documents

The Student Code of Conduct.

Plagiarism Penalty Grid

Step 1: Assign Points Based on the Following Criteria

1 History

1st Time	**100 points**
2nd Time	**150 points**
3rd/+ Time	**200 points**

2 Amount/Extent

Below 5% OR less than two sentences	**80 points**
As above but with **critical aspects*** plagiarised	**105 points**
Between 5% and 20% OR more than two sentences but not more than two paragraphs	**105 points**
As above but with **critical aspects*** plagiarised	**130 points**
Between 20% and 50% OR more than two paragraphs but not more than five paragraphs	**130 points**
As above but with **critical aspects*** plagiarised	**160 points**
Above 50% OR more than 5 paragraphs	**160 points**
Submission purchased from essay mill or ghostwriting service	**225 points**

** Critical aspects are key ideas central to the assignment*

3 Level/Stage

1st year	**70 points**
Undergraduate (not 1st or final year)	**115 points**
Final year/Postgraduate	**140 points**

4 Value of Assignment

Standard assignment	**30 points**
Large project (e.g. final year dissertation, thesis)	**115 points**

5 Additional Characteristics (to be used only in extreme cases)

Evidence of deliberate attempt to disguise plagiarism by changing words, sentences or references to avoid detection: **40 points**.

Step 2: Award penalties based on the points

6 Summative Work

In all cases a formal warning is given and a record made contributing to the student's previous history.

Points	Available Penalties (select one)
280-329	• No further action beyond formal warning • Assignment awarded 0% - resubmission required, with no penalty on mark
330-379	• No further action beyond formal warning • Assignment awarded 0% - resubmission required, with no penalty on mark • Assignment awarded 0% - resubmission required but mark capped or reduced*
380-479	• Assignment awarded 0% - resubmission required but mark capped or reduced • Assignment awarded 0% - no opportunity to resubmit
480-524	• Assignment awarded 0% - no opportunity to resubmit
525+	• Case referred to Discipline Committee

7 Formative Work

280-379	• **Informal warning**
380+	• Formal warning, with record made contributing to the student's

	previous history

* Normally, marks will be capped at the pass mark for the assignment.