School of Engineering Academic Integrity Guidelines

Richard Hughey
October 1999

1 Background

Unfortunately, academic dishonesty at UCS is far more common than it should be, especially on homework and programming assignments — many students, including some good ones, view copying simply as a risk of getting a zero on an assignment. An aggressive School-wide effort to sanction the academically dishonest will get the word out as to what is and is not acceptable.

This issue is discussed in Dent Earl's Fish Rap Level article of May 5, 1999, Cheaters Prosper: Trouble in the CS department:1

"Before tests," he said, "you talk to your friends figuring out where you will sit before you’re going into class, what hand they write with, which ones wear glasses, how long their hair is, who’s going to wear a hat."

1 A computer science major claims to have seen and heard about various forms of cheating, from the ubiquitous code exchanging to a more serious transgression. “One student was working on a lab computer and all of their code was there,” said Morrows. “Another student came in while the author of the code was away and stole all the code and submitted it. The student who authored the code then returned and submitted his work and got [accused of cheating].”

If you have not done so, please read the Fish Rap Level article and Kevin Karplus’ response2 — they are quite informative. This sheet includes an overview of University and SOE policy and a description of available code comparison software.

The primary responsibilities for TAs and readers are to be vigilant, assist in proctoring examinations, and run code comparisons of current classes and past classes. Once possible academic dishonesty has been found, the TA should report the incident to the instructor and assist in the instructor’s investigation as requested. The TA should not bring the issue up with the student.

2 Definitions

Most academic dishonesty is a form of plagiarism. Webster's Seventh Edition tells us that to plagiarize is: to steal and pass off as one's own (the ideas or words of another) to present as one's own an idea or product derived from an existing source. There are important things to remember about citation and quotation.

Citation is required whenever the source helped you form your ideas, you presented someone else's work, or the reader might be interested in the other work. Repeat the citation if specific facts, tables, or figures are presented. You can use citations to "personal communications" for a proof someone showed you at the whiteboard. You should mention everyone who helped you in even the most minor way, either by citation or by acknowledgment. Be sure to include acknowledgment to any granting or fellowship agency that supported you or any of your coauthors.

Quotation, in the form of quotation marks or indented text, is required whenever text is an exact copy of text from a source. A citation is also required.

3 Instructor Responsibilities

Instructors must follow the University policy. School of Engineering guidelines are required to ensure that our internal records of academic dishonesty events are maintained. This will involve enlisting your teaching assistants and readers to help detect academic dishonesty.

3.1 University Policy

University policy is discussed in the UCSC Academic Integrity Policy. This policy was adopted in Fall 1999.

For undergraduates, the instructor's guide to academic dishonesty is maintained. This guide is available online for students to consult.

By failing an exam, a student may be subject to any grade given for the course, including an outright failure. For both undergraduate and graduate students, regardless of the grade received, the student may be subject to any grade given for the course, including an outright failure.

The initial responsibility to reduce the potential for cheating is to reduce the potential for cheating. Possibilities here include:

1. Announcements at the beginning of the quarter that cheating will not be tolerated and may lead to serious consequences, such as failing the course, or being suspended or dismissed from the University.
2. Clarification of ambiguous areas — where collaboration ends and cheating begins.
3. Using picture IDs in large classes to prevent stand-ins.
4. Maximizing seating distance during examinations and attentively proctoring examinations.
5. Disallowing baseball caps, sunglasses, and the like.
6. Allowing a page notes (or complete open book) will eliminate one behavior as being cheating.
7. Once cheating or suspected cheating has occurred, instructors should refer to the Academic Integrity Policy for the complete procedure. The first step is to present the student with the student's provost on a standard form.

Instructors must report all cases of suspected cheating to the student's college provost within 3 working days of meeting the student. There is a standard form for reporting incidences. In addition to reporting the incident, instructors have several course-related options:

1. Additional work
2. Re-examination
3. Exclusion from course
4. Change of grade
5. No credit for work in question
6. Exclusion of incident in student's narrative evaluation

At present, there are no standard SOE guidelines. I personally feel that cheating on a quiz or exam should result in exclusion. In my classes, code copying can range from exclusion to no credit on the work in question depending on circumstances (e.g., direct copying or a collaboration that went too far).

If you and the student disagree about whether or not academic honesty took place, do not give any grade to the student until after hearing back from the Provost. If you expect to be away, leave information with your department chair about what grade should be assigned based on whether or not academic dishonesty was found.

When you receive notice from the Provost, be sure to put a copy in the student's file.

1http://pitr.ucsc.edu/ftn/ngen/0.13/checker/prosper.html
2http://pitr.ucsc.edu/ftn/ngen/0.14/letter014.html

Never present text that is a minor rewording of someone else's work; it cannot be properly quoted, yet it is not your work. Do not compose material while looking directly at others' work.

Code swapping and copying on examinations are plagiarism if not acknowledged. Copying on examinations is always cheating. Acknowledged code swapping may or may not be cheating depending on the course.

3.1 University Policy

University policy is discussed in the UCSC Academic Integrity Policy. This policy was adopted in Fall 1999.

For undergraduates, the instructor's guide to academic dishonesty is maintained. This guide is available online for students to consult.

By failing an exam, a student may be subject to any grade given for the course, including an outright failure. For both undergraduate and graduate students, regardless of the grade received, the student may be subject to any grade given for the course, including an outright failure.

The initial responsibility to reduce the potential for cheating is to reduce the potential for cheating. Possibilities here include:

1. Announcements at the beginning of the quarter that cheating will not be tolerated and may lead to serious consequences, such as failing the course, or being suspended or dismissed from the University.
2. Clarification of ambiguous areas — where collaboration ends and cheating begins.
3. Using picture IDs in large classes to prevent stand-ins.
4. Maximizing seating distance during examinations and attentively proctoring examinations.
5. Disallowing baseball caps, sunglasses, and the like.
6. Allowing a page notes (or complete open book) will eliminate one behavior as being cheating.
7. Once cheating or suspected cheating has occurred, instructors should refer to the Academic Integrity Policy for the complete procedure. The first step is to present the student with the student's provost on a standard form.

Instructors must report all cases of suspected cheating to the student's college provost within 3 working days of meeting the student. There is a standard form for reporting incidences. In addition to reporting the incident, instructors have several course-related options:

1. Additional work
2. Re-examination
3. Exclusion from course
4. Change of grade
5. No credit for work in question
6. Exclusion of incident in student's narrative evaluation

At present, there are no standard SOE guidelines. I personally feel that cheating on a quiz or exam should result in exclusion. In my classes, code copying can range from exclusion to no credit on the work in question depending on circumstances (e.g., direct copying or a collaboration that went too far).

If you and the student disagree about whether or not academic honesty took place, do not give any grade to the student until after hearing back from the Provost. If you expect to be away, leave information with your department chair about what grade should be assigned based on whether or not academic dishonesty was found.

When you receive notice from the Provost, be sure to put a copy in the student's file.

1http://pitr.ucsc.edu/ftn/ngen/0.13/checker/prosper.html
2http://pitr.ucsc.edu/ftn/ngen/0.14/letter014.html
3.2 School of Engineering Policy

Copies of the required form to the college provost (or Dean of Graduate Studies), should also be provided to the student's faculty advisor and the SOE's student records.

The School of Engineering is adopting a policy of major disqualification based on academic dishonesty. Thus, it is vital that we maintain accurate records of student transgressions. The college provosts are not as close with their information; we cannot rely on receiving this important information from external sources. It is also possible to write up cheating incidents in the narrative evaluation. This could be an effective threat at the start of the course.

Several CE faculty require that collaborative work, when allowed, be acknowledged.

3.3 TAs and Readers

Instructors should directly discuss cheating with TAs and readers. TAs and readers have two responsibilities: observation and reporting. Readers should look for similar assignments that may indicate copying, and TAs should watch for cheating during a test. After observing a suspicious incident, the TA or reader should bring the matter directly to the attention of the instructor, not the student involved.

4 Code similarity checking

There are several code similarity checkers that are readily available. The one that I am most familiar with was written by Dick Grune, Vrije Universiteit, Amsterdam, and is called SIM6. The program is about 10 years old, and has recently been updated to include a Java parser. The program is close to linear in the number of programs that it is searching. It builds up a large hash table based on the parse trees of the input programs.

Another code comparator called JPLAG6 by Guido Malpohl is available for downloading and WWW use, and MOSS6 by Alex Allen is available for WWW use. The plagiarism.org site can compare papers, as discussed in TeckWeek4.

The sim program is difficult to use because its output consists of the sequence of matching segments. The most closely matching segment is printed first, followed by the second most closely matching pair, and so forth. Because two similar programs may have a large number of matching segments, the important metric is not being on the top of this list, but having a large coverage on the list. The V.U. has a series of scripts for their local situation by Matty Hustgens. These were not as useful for our situation, but provided several important ideas for the development of our own parsing scripts.

The scripts work as follows. First, the appropriate sim program is run on the assignment or class as a whole, preferably including the work of previous quarters as well. If students are handing in multiple files, you may wish to concatenate them together, excluding any common code distributed to the class (or including it, and adjusting your thresholds upward). Second, the resulting sim file is processed by the parseerror script. This program will combine the results of matching segments for each file. Any pair of files that exceed a certain threshold will have a pairwise similarity comparison performed, producing a side-by-side listing of the matching segments. The result of this is a ranking of complete files by their degrees of similarity. The file can be viewed with HTML to provide quick access to the comparison files and the source code.

Third, after the instructor has identified which cases appear to be academic dishonesty, the parseerror script can be used to send email to the students asking them to come to your office hours, generate a report form for sending to the college Provost, and create a 4-up version of the source files and the code difference for sending to the Provost.

4.1 Initial Similarity Comparison

The programs and scripts are available on the cse machines in /cse/faculty/rph/sim/. Documentation is available in html8, postscript, pdf, and dvi. Because I have no desire to install or maintain them on the CATS machines, you will need to copy the assignments to our file system. This is good for two reasons. First, it makes the code comparator unavailable to people on the cats machines. Second, in the process of copying over the homework assignments, you will be able to archive them in the /cse/classes directories — many of the /cse/classes directories have an ARChive subdirectory, /cse/classes/csc4012c/ARCHIVE, which is designed for performing this work. This is vital because copying from one quarter to the next is one of the most common occurrences. It is well worth the effort to get copies of previous assignments from the course file.

To copy files over, you can use the following commands from a CATS machine (here, 'bin' puts FTP in binary mode):

```
tar -cf - > .sim after compress > hw.tar.Z
ftp -a rph
Password: xxxxxx
put hw.tar.Z
put sim.out
```

Once you have copied the tar file over, you can do the following:

```
nmake this_quarter
cd this_quarter
scat /hw.tar.Z | tar -xf
```

The next thing to do is to make the first pass with the similarity comparator. There are several possibilities for this, some of which are illustrated in the runsim script. I find it useful, especially when comparing with old directories, to create a directory specifically for the similarity comparison. In this directory, make a symbolic link to the current quarter's work, make a subdirectory called 'old' and make links in that subdirectory to the previous quarters. The sim scripts will look for 'old' in path names and compare email accordingly. The scripts will never perform a one-on-one comparison between two files with the same login ID.

If Unix command line limits are not exceeded, one can do everything all at once, with a command such as:

```
sim.c = -c /cse/courses/cse4012c/old/sim.out/sim.out
```

Here, the two switches tell sim to not compare a file with itself, and to only output the scores of the matching segments, rather than the segments themselves. Additional options are listed on the sim man page11. The next arguments are the new files to compare, followed by a slash, and a list of previous files. The previous files will not be compared with each other. The sim programs include sim.c, sim.java (for C++ as well), sim.w, sim.hxx, sim.s, and sim.teact. New parsers can be created as well.

The Perl scripts make use of 'old' and 'samples' being in the path names. Samples are used to (through an imperfect transitive closure procedure) weed out acceptable code copies. The 'old' indicator is used in computing email and reports.

For a very large class, this command line can exceed the number of argument limits of Unix. If this is the case, you will want to compare each assignment on its own and cut the results together. (You can either cut all the sim.out files together or, if you would like to keep the assignments separate, the sim.worst file, and files discussed in the next section.) There are examples of this in the runsim script. If you have late handin directories, be sure to compare these as well; undoubtedly, several of the late submissions will be based on on-time work.

It is critical to review previous quarters' work, both yours and for other instructors, to find assignment overlap. With the expansion of /cse/classes, there is now room to archive prior work. Please include a README for each quarter that has a brief description of what each assignment was.

---
4

---
6http://www.vu/vulj/dick/sim.html
7http://www.acs-biotech.de/~uniri/Uniri/.
8http://www.CE.Berkeley.EDU/~Allen/moss.html
9http://www.isiway.com/articles/1-10-2000/clserv.htm
4.2 Parsing the SIM output

The following command will parse the raw output of the sim program.

    parseim < sim.out.

You can find out about this script's options by running it with -h help. The options include:
- lang: The language to be used. The default is en; others include java, spin, lisp, m2, pacc, and text.
- simroot: Root name of sim program. Default is /ce/facek/fyp/sim/sim.
- pairlevel: Absolute number of token matches required for a 1:1 comparison (that may be later discarded).

Default is 2 tokens.

frac and level: If both the frac percentage of tokens match and the absolute level number of tokens match, a one-on-one comparison is recorded. Default values are 30% and 100 tokens.

allfrac: Whenever allfrac or higher percent of tokens match, regardless of level, a one-on-one comparison is recorded. Default value is 30%.

samplefrac: Percent of tokens that match to drop a key that matches a key with 'samples' in its name.

Default value is 60%.

level: Minimum token score to keep a 1:1 comparison. This will prevent, for example, 'hello world' programs with 60% matching from being compared.

alllevel: Whenever alllevel or higher tokens match, regardless of frac, a one-on-one comparison is recorded.

Default value is 200 tokens.

mixrep: Minimum number of one-on-one reports as a percentage of number of files, regardless of thresholds.

Default is 5%.

maxrep: Minimum number of one-on-one reports as a percentage of number of files, regardless of thresholds.

Default is 30%.

title: A title for the run, used in the HTML file.

The parseim program will create a pairwise comparison whenever its thresholds have been met. This comparison will be named something like:

    60_file1.com,60_file2.com,60_sim2.com.

The components are the percentage of matching tokens (60), the two user IDs of the students, and an extension. This file will be stored in one of the two directories of the source files. On completion of all academic dishonesty cases, you will want to perform a command such as:

    find -o *name '*sim' -exec rm {} \;

to remove all these comparison files, which is what the clean_sims script does, as well as remove any

    sim.err, sim.summary, sim.diff.4.ps, and sim.rep.ps files.

The parseim program will look up student names in a local copy of the CATS passwd file, which I hope to maintain up-to-date. If an entry is not found in the passwd file, a failure command is used.

The parseim program also produces two other files: sim.summary and sim.worst. The first file of these files is a log of the script execution. It includes the sum of scores from the initial run for each file, as well as as the names of the pairwise similarity files that it decided to generate. Most likely, you will not look at this file. The second file is presented in two forms: sim.worst and sim.worst.html. The best way to use this file is to bring it up in your favorite Web browser as, for example,

    /bin/9m -f sim.wall
/bin/9m -f sim.wall.html
foreach i (cat http bu6 bu6 b6)
        $sim -a -o sim/$i/1*?c_sim/$i/1*?c o/sim/$i/1*?c > sim.out
$parse -title "$i" $i < sim.out
cat sim.worst > sim.wall
cat sim.worst.html > sim.wall.html
end
/usr/bin/mv sim.wall sim.worst
/usr/bin/mv sim.wall.html sim.worst.html
$worst < sim.worst

Figure 1: Sample email (from sim.worst) generated by parseworst.

This script cycles through 5 homework directories, comparing the given homework assignments against old ones. One problem with this approach (required by the limit on command-line arguments) is that if two similar assignments were renumbered between quarters, the problem will not be discovered. The sim.wall files are used to accumulate the output files of parseim, and then copied back into the sim.worst.

For each pairwise comparison, this script provides the score, a link to the pairwise comparison, the student names and email addresses, and links to the source files. Additionally, there is a check box associated with each pairwise comparison. By default, each of these check box is off. If, after examining the pairwise comparison, you believe that copying has taken place, click on the check box to turn that case on.

Examining this list may give you some insight into the proper threshold settings for the programs. For example, if the lowest scoring pair on the list exhibits clear similarity, you should re-run the parseim program with lower settings.

Once you have selected which cases are suspicious, enter your login ID at the bottom of the form and press the "create a key file" button. This will cause a list of keys to be emailed to you, which you can then save in a file for use with the next script. Call this file sim.keys, and if you select keys more than once, be sure to delete the previous file before saving the new one to avoid appending the new file to the old file. You do not need to remove email headers from the file. Alternatively, if you do not provide an email address, you can simply cut and paste the keys into a file in the appropriate place.

4.3 Informing Students and Provosts

The parseworst script allows you to generate email, reports, and documentation of your academic dishonesty cases. You can run this program with a command similar to:

    parseworst -check -office "Wednesday 10-12" -keys sim.keys < sim.worst

This example illustrates several options of the script. The first option will create sample email (Figure 1) to be printed to the file sim.email. The next option specifies the office hours that will be included in the email. The keys option specifies that only those pairwise comparisons listed in the sim.keys file should be processed.

Whenever the parseworst script is run, it will generate a sim.1st file. Rather than ranking cases by percentage of similarity, this list file will list cases alphabetically by student name. Individual pairwise similarities will thus be listed twice in all cases. Students who took part in groups of three or more collaborators
Figure 2: Sample form (from sim.forms) generated by parseworst.

may have several entries for the same program. Associated with the sim.list file is another HTML file, sim.list.html. As with the sim.worst.html file, check boxes are present to further refine the key file. As mentioned above, be sure to delete the old key file before saving the new one.

The options for the parseworst script can be seen by running the script with an invalid option. The options include:

- **email** send electronic mail to all the students involved (Figure 1). If the student has work from a previous quarter that is being copied, this will be noted in electronic mail. The mail will also ask that the student attend your next office hours. Always use the check option before actually sending the email. A copy of the mail will be sent to you as well.

- **check** Print the electronic mail to sim.email rather than sending it (Figure 1).

- **diff** Create a sim.diffs file and a sim.diffs.ps file of the pairwise similarities. In general, this option is not needed because of the HTML interface.

- **report** Place complete documentation (the pairwise difference and the two source codes) in a 4-up PostScript file, sim.rep.ps. You will need to include a copy of these reports when you send documentation to the Provost. The reports for all students will be in this single file.

- **form** Create a sim.form that has ASCII one-page forms (Figure 2). These forms include as much as sim knows about the case, as well as places for you to fill in the details of your meeting with the student and academic sanctions that you have decided on. There is a place for both you and the student to sign after your meeting (whether or not the student agrees that academic dishonesty occurs). You'll want to print these out, possibly copying the forms before your meetings with the students so that you can provide them with a copy immediately.

caredir: Create a directory containing one subdirectory per student, each with files form, email, and report.ps. This option overrides --form, --check, and --report, and disables --email. This is an excellent to store cases. ASCII versions of the code are not copied, so you will still want to archive the entire class. If you use the check option, you can then edit the individual email files as appropriate and do something like the following, where the -t option to sendmail specifies that header lines should be taken from the message itself:

```bash
for each i (caredir/*/email) /usr/lib/sendmail -t < $i
```

The following options all include an argument of some sort or another:

- **keys** Specify a file of keys. Only cases that are included in this file will be processed.

- **prof** Specifies the professor's name. Usually, this is successfully looked up from your finger information.

- **profemail** Specifies your email address. This is proxied to the result of the whoami command.

- **office** Your office hours, optionally used when sending email.

- **phone** Phone extension, for use in creating forms.

- **class** The name of the class. Include both the title and the call number. This is required for sending email and for creating forms.

- **quarter** The quarter in which your class took place, for use in creating forms.

- **title** Your title. This will be used in creating forms. It defaults to "Computer Engineering and Computer Science".

- **far** A narrative evaluation header file. This will be used to extract major and college information about your students for the form. The matching is done on the email address, so if the file lists 'unknown' rather than 'student@xsc.edu', you will have to fill in the fields by hand.

- **pages** The number of pages on a page to be used in creating the documentation file. The default is 4, but nine is also readable.

As mentioned above, before meeting with a student you'll want to print out a copy of the report and the form. Additionally, it would be a good idea to check with the undergraduate advising office as to whether or not any of the students are (or so far as we know) repeat offenders.

When you meet with the student, fill out the form with the student. If you decide that academic dishonesty did not take place, discard the form.

If you and the student agree that academic dishonesty did take place, decide upon an academic sanction and suggest any disciplinary action that you feel would be appropriate to the Provost. You may complete your course reports and narrative evaluations without waiting to hear from the Provost.

If you and the student disagree, discuss the disagreement in the comments section, and ask the student to sign the form.

Once you have completed the form, be sure to make copies for the student, yourself, the undergraduate advising office, and possibly the chair of the student's department. Send the original form with the documentation to the college provost within 3 working days of your meeting. With luck, you will hear back from the provost within a few weeks. If you do not, a phone call or a mail message would be appropriate. Under current policy, the Provost is required to notify you in writing of his or her opinions and decisions on the matter.

### 4.4 Directory organization

Please archive past quarter of material in /cse/classes/cmp3609/archive/old/./quarter. If you put your new material in, for example, archive/fall, and add a link to the old directory, you can put the run script in that directory and modify it for running the program. At the start of the quarter, check with the previous instructor about whether or not any of the old .sim files are required. Run cleanall, move the previous order to the old directory, and then create a directory for your new quarter. Be sure the protection is set so that the files are owned by the appropriate class-specific group with rwx, and that there is no world access.

In each directory, keep a README file for each homework briefly mention what each assignment was. It is critical to keep these records so that future instructors and teaching assistants will not what previous laps
to compare against. A recent version of CS101 had several people handing in disjoint set code from several quarters previous.

4.5 Final thoughts
To re-emphasize, it is critical to provide a copy of the form, as well as any response from the Provost or Graduate Dean, to the undergraduate or graduate advising office. Even the most minor case, or the most repentant student, may be a repeat offender or may become a repeat offender. Without the formal documentation that you are creating, it is impossible for a Provost or the Graduate Dean to treat a second occurrence as a second occurrence.