USING AN EFFECTIVE GRADING METHOD FOR PREVENTING PLAGIARISM OF PROGRAMMING ASSIGNMENTS

C. Jinshong Hwang
Department of Computer Technology
Purdue University
W. Lafayette, IN 47907

Darryl E. Gibson
Department of Computer Sciences and Mathematics
DePauw University
Greencastle, IN 46135

I. Introduction.

Cheating on programming assignments, it would seem, has become a way of life for many, and fraudulent misrepresentation of one's credentials for entrance into the marketplace is increasingly commonplace -- particularly when the monetary and social rewards are very attractive, or at least perceived as being so. [2,4,5] The computing industry at present does indeed offer many attractive incentives.

It is not the purpose of this paper to sermonize on the evils of cheating and the effects of cheating upon the individual character. (We simply assume that cheating on programming assignments is a highly undesirable practice which has adverse effects upon the students' prepartation for their later professional performance in the computing industry.) There are many ways used to prevent plagiarism on programming assignments. Some of them deal with moral standards, some with cheating policies (disciplinary) [4,5], some with threats, and some with detection software [1,2,3,6], but this article deals with grading methods.

The two main purposes of this paper are: 1) to discuss four commonly-used grading methods (which we shall call methods A, B, C, and D) employed with programming assignments and 2) to present by way of recommendation two experimental methods (which we shall call methods X and Y) which support our thesis that positive prevention of cheating on programming assignments through the use of an appropriately-designed grading method is far more effective than the other approaches in general use.

To accomplish these two purposes we present first (in Section II) a list of the most common techniques employed by students for cheating on programming assignments, a list of the most common instructors' responses to this type of cheating, and a brief discussion of the general seriousness of this type of cheating, not just as a problem in this country but as a problem of international scope. Section III is devoted to a discussion of each of the four general methods A, B, C, and D, together with a description, their advantages and disadvantages, and our conclusion and recommendation concerning them. In Section IV, we describe the two experimental methods X and Y which are strongly recommended as grading methods. Finally, in Sections V and VI, we offer our summary and overall conclusions.

II. Scope and Seriousness of Plagiarism on Programming Assignments.

It is assumed that preparation of the student for a career in the computing industry must include both factual, theoretical knowledge of the discipline as well as certain practical, applied skills. All educators in the field recognize this fact, but the relative proportion of theory to practice often varies, depending partly upon the instructor's view of the relative importance of these two aspects of discipline, partly upon whether or not the student is

actually doing the work, and partly upon the amount of effort an instructor wants to put into producing a grade (objective-type tests are often easier to grade than programs). In this paper we are concerned only with the second point: Are the students actually doing their own work on programming assignments? To what extent can the instructor trust the students to work independently and honestly on programming assignments?

Students have devised an assortment of ways for cheating on programming assignments. Below, are listed several of them:

1. Copying a program by changing only the author's name.
2. Having someone else write all or part of the program.
3. Copying a program given in an earlier class.
4. Copying a program by changing only the line numbers.
5. Copying a program by changing the documentation.
6. Copying a program by changing the logic a little.
7. Copying a program by changing the variable names.
8. Copying a program by changing the logic a lot.

In response to these and other forms of cheating on programming assignments, several methods have been developed. Some of them emphasize the detection of violations, while others stress the prevention of cheating. Some methods use negative reinforcement; others use positive reinforcement.

1. Set up a punishment policy to discourage students from cheating. [4,5] This method is ineffective, since it is essentially negative. Students can not be expected to react positively to a negative approach. It can only achieve minimal results.

2. Set up software detection. [1,3,6] Ingenius, sophisticated techniques have been created to spot instances of computer plagiarism. It is doubtful that such methods can be contrived to catch every type of cheating. Such expensive systems are probably only useful as interesting academic exercises.

3. Raise the consciousness of the students to understand and appreciate what they must know in order to obtain a degree in the discipline and later to function effectively as computing professionals. This is essentially a positive approach, appealing to the student's higher instincts. But it is generally ineffective with most students who, although they realize the need for adequate preparation for their future careers, are nevertheless mainly interested in the here-and-now matters of passing tests and courses.

4. Inform students that they may be called into the office at any time to verify what they "claim" to have learned on programming assignments. This is basically a cynical approach, announcing in a loud and clear voice to the students that the instructor does not believe or trust them. This method is ineffective and is apt to invite confrontations with students who are called upon to defend themselves.

5. Assign grades according to the ratio of programming assignments and exams. This is the method we support in this paper, believing that positive prevention is much more effective in the long run than the kinds of prevention and detection suggested by the other methods. We feel that most cheating can be prevented in a positive way and that most negative measures are not only ineffective, but they also breed further mistrust which in turn interferes with the teaching-learning process.

It is just as difficult for institutions as it is for individuals to admit that cheating is going on, so we, personally and institutionally, strive continuously to maintain the faCade of untarnished respectability -- but at what cost? What are the

ultimate economic costs, for example, of low per capita productivity in a business or industry caused by incompetent employees who were hired using fraudulent credentials, who claimed on their college transcripts that they had taken such and such courses in computer science or computer technology but who in fact can hardly program or program well? How much does it cost the company, while the employee learns (on the job) what should have been learned before the person was hired?

These problems are not unique within this country. Every industrialized country makes extensive use of computer technology, and many people, the world over, are attracted similarly to the computing industry by among other things, the promise of relatively high wages. Hence, the motivation for cheating and the temptation to cheat on programming assignments at the preparation level is universal.

Much of this problem, certainly not all of it, could be alleviated if we who are charged with the preparation of future computing professionals could guarantee, to the extent possible, that our graduates have in fact learned the material and are in fact competent to step into the marketplace. We can do this only if we can design and adopt practices which will systemmatically require the students to master both the theoretical and practical aspects of the discipline. We now turn to a discussion of several methods used in assigning grades on programming assignments. With each method we shall offer a brief description, the methods's perceived advantages and disadvantages in terms of the cheating on programming assignments. Finally, we present our conclusions and recommendations for dealing with cheating effectively and positively on programming assignments.

III. General Methods Used to Cope or Not to Cope with Cheating.

A. Exams (including routine quizzes) weighted proportionately heavier than programming assignments.

For example, exams may be given a weight of 70% and the programming assignments a weight of 30%.

The instructor may wish to weight exams and programming assignment approximately equally, but feels that since cheating is going on, a heavier emphasis on exams will give him more control over the grade-producing process (which cheating tends to take from him). There are other reasons for an instructor's placing a heavier emphasis upon exams, but we are interested only in those which are related to cheating.

Advantage

1. If students do not really understand the principles involved in the programming assignments, they cannot do well on the exams, having a significant number of program-related questions on them.

Disadvantages

1. Honest students are apt to spend disproportionately more time on the programming assignments than on preparing for the exams.
2. Students can concentrate on the text and lecture notes and, consequently, do well on the exams. They can pass the course with a minimum of programming practice.
3. If students happen to have a bad day, they may do poorly on the exam.

On the positive side, the students have to know the material in order to do well on the exam. This means that the grades the students receive should be a true reflection of their work. But when we weigh the negative results of this method against the positive ones, we find that we cannot recommend it. The honest students, in a sense, are penalized for working

hard to learn programming skills and also by having to take a chance on having a bad day on the exam, while students weaker in programming can prepare mainly for the exams and get a high enough score on it to offset whatever they got on the programs to pass the course--without having learned any programming skill.

B. Programming assignments weighted proportionately heavier than exams (including routine quizzes).

For example, programming assignments may be given a weight of 70% and exams a weight of 30%.

The instructor who uses this method probably feels that programming techniques are relatively more important than theoretical knowledge. This instructor may also (perhaps naively) feel that he doesn't have a programming assignment cheating problem in his class.

Advantages

1. This methods encourages students to learn by doing. Students are less inclined to write programs by simply emulating syntax rules.

2. Programming assignments are often time-consuming. The grade obtained following this method, therefore, is proportional to the time and effort expended.

3. This method is good for students, if students do not cheat on programming assignments.

Disadvantages

1. Students can copy programs and pass the course with minimal work.

2. This method encourages students to copy other students' programs.

3. This method may encourage better students to neglect studying the theoretical material.

On the plus side, students are given ample opportunity to develop programming skills, and the grade received should represent their work. On the other hand, the better or honest students may end up slighting the theoretical subjects. The weaker or dishonest students are tempted to copy the work of others. And since so much of the grade is dependent upon programming, the incidence of cheating increases as the programming assignments get harder. We do not recommend this method.

C. Exams and programming assignments weighted approximately equally.

For example, each might be given a weight of 50%. Clearly, this method is a compromise between the previous methods. It is an attempt to recognize the relatively equal importance of theory and practice. This is a widely-used and accepted model.

Advantage

1. This method encourages students to approach the exams and the programming material with equal seriousness.

Disadvantage

1. If students cheat on the programming assignments, they can still get an average grade. For example, if they get only a 40% on the exam but cheat and get an 100% on their program assignments, their average is 70%.

Because this method encourages the students to master both the theoretical and applied aspects of computing, it has a lot going for it. Still, it is possible to cheat "well enough" on the programs to offset possible low marks on exams in order to pass the course without learning the important programming skills.

D. Final exam used as evidence of what the student has learned.

If the students receive a passing grade on the final exam, they receive the grade earned throughout the semester; including the final exam grade. If, however, the students fail the final exam, they fail the course. A failing grade on the final exam is interpreted as evidence of cheating throughout the semester.

Advantages

1. Students will really try to understand the material in order to do well on the final. This method encourages students to do programming assignments regularly.

2. If the final exam does represent the students' understanding, this is a good method.

Disadvantages

1. During the final exam period students have many pressures and distractions from other courses. It is difficult for students to put their full, undivided attention on the exam.

2. Students might have a bad day on the exam. Consequently, all, or most, the time and effort for a semester could be nullified.

3. If final exam problems do not represent material fairly, this method can be a disaster.

4. This method has a very high potential for triggering confrontations between the students and the teacher, when the exam does not properly test what was taught and what was learned.

In principle this method sounds like a good one, and it is if the final exam fairly represents the programs the students were asked to produce throughout the semester and if the students have a good unobstructed day on the exam. We have some control over the first condition (although manufacturing such an exam would be a monumental undertaking), but the second condition is almost completely out of our hands and to a large extent out of the hands of the students. If the first condition is not met, and the exam is not a fair one, we are sure to invite a confrontation with the students. And if the second condition is also not met, we are putting our necks on the chopping block. We cannot recommend this particular method of suicide. Even if the final exam does represent what was taught and what was learned, an effective grading method for the semester is still desirable.

E. Programming Assignment-Related Quiz Associated with each Programming Assignment.

This method is similar to method C, in that it also recognizes the equal value or importance of theory and practice, by assigning approximately equal weights to each of these areas. This method goes a step further, however, in providing a concrete mechanism for encouraging students to do the programming assignments. This method assumes that the students, acting in their own self-interest, will realize that copying someone else's work will not prepare them for the programming assignment-related quiz and will, thus, act accordingly.

The program assignment related quiz, which is given on the due date of the programming assignment and which normally takes about twenty minutes, asks the students to reproduce a small segment of the programming assignment which deals with some particular programming principle, logic and design involving the completion of programming assignments. There are two variations of this method which we shall discuss in the next section.

The small programming assignment related quiz should represent the programming assignment completely. That is, the questions appearing on the quiz should deal exclusively with the principles and concepts covered in the programming assignments and that extraneous, non-relevant material should be avoided. The objective of the quiz is to test the student's knowledge of the idea behind the programming assignment only. This means that both the programming assignment and programming assignment-related quiz must be carefully thought out and correlated one with another.

IV. Experimental and Recommended Methods

Over the past several years we have used some of the general methods described above (methods A, B, C, and D) with varying degree of success but were never satisfied that we (or our colleagues) were really in control of the situation, that some of the most important consideration in the learning processes and in preparing, giving, and evaluating programming assignments were being overlooked. This dissatisfaction prompted us to experiment with new methods. This section is devoted to discussing the two methods which give us a higher measure of assurance that our students are actually learning the material and upon which we have to base our grading decisions. We shall refer to these methods as method X and method Y.

In each of these two methods there are two factors to take into account: the score obtained on the programming assignment and the score on the programming assignment-related quiz. The "provisional" score is the initial score on the programming assignment, before it is modified by the score or percentage on the programming assignment-related quiz. The "actual" score is the final score on the programming project after the modification.

X. Percentage on programming assignment-related quiz applied to the score obtained on the programming assignment.

```
      Provisional score on programming assignment
  X   Percentage on programming assignment-related quiz
-------------------------------------------------------
  =   Actual score on programming project
```

For example, there is a total of 100 points possible on a programming project. If a student gets 80 points (as a provisional score) on the programming assignment and 90% on the programming assignment-related quiz, the student's actual score on the project is 72 points:

72 points = 90% X 80 points

Advantages

1. This method encourages the students to do the programming assignments in order to do well on the programming assignment-related quizzes.

2. The score actually represents the students' understanding of the programming assignment.

3. The grade is proportional to the time and effort expended.

4. This method offsets the disadvantage of methods C and D.

Disadvantages

1. If the students have a bad day, the score will not represent their true ability.

2. If the programming assignment-related quiz does not represent the programmng assignment well, the score does not represent the students' ability.

Even though we have very little control over the kind of day the students may have on the exam, we do have control over the type of programming assignments and the associated quizzes we give. If we control these factors, this method is an excellent one, one which we highly recommend.

Y. Score obtained on the programming assignment-related quiz added to the provisional score obtained on the programming assignment.

```
     Provisional score on programming assignment
+    Score on programming assignment-related quiz
-------------------------------------------------
=    Actual score on programming project
```

For example, there is a total of 100 points possible on a project, 50 points for the programming assignment and 50 points for the programming assignment-related quiz. If the student gets a provisional score of 40 points on the programming assignment and 45 points on the quiz, the student's actual score on the project is 85 points:

85 points = 45 points + 40 points

Advantages

1. This method encourages the students to do the programming assignments in order to do well on the programming assignment quiz.

2. The score actually represents the students' understanding of the programming assignment.

3. The grade is proportional to the time and effort expended.

4. This method can offset the disadvantages in method X to a minimum.

5. This method takes care of all the disadvantages associated with all the general methods.

6. This method represents the students' grades very well for all unexpected situations.

7. This method is well accepted by all categories of students.

Disadvantage

1. This method still has the disadvantages of method X, but it can offset the disadvantages to a minimum.

This method is not perfect, but it will take care of the problems encountered in the discussion of all the previous methods. This is the method which we most highly recommend according to our experience and experiment. Students' responses as well as instructors' reaction are very favorable in method Y.

Comparing each of the grading methods discussed in this paper side by side reveals some most interesting results. For the sake of this comparison we shall assume that a student cheats on all programming assignments and receives full credit for them. We shall also assume, that because the student does not understand the material, he or she receives only 20% on the exam, quizzes, and so on. With each of the different methods, he or she will receive the following grades:

Method A	42%
Method B	76%
Method C	60%
Method D	20%
Method X	20%
Method Y	40%

There is indeed a great difference among the grades! We observe that methods B and C (with grades of 76% and 60%, respectively) allow the dishonest students to pass. These methods are clearly too lenient on the dishonest students. With methods D and X, we know that not only the dishonest students but also the honest students are being penalized unfairly if they have had a bad day. The resulting grades of methods A and Y are similar, but method A has too many disadvantages. Method Y represents the compromise among all of the discussed methods: It not ony prevents the dishonest students from passing and rewards the honest students, but it also contains all of the advantages of the other methods. Needless to say, we recommend method Y most highly.

V. Grade Distribution Using Method Y

According to the method which we use for grading programming projects, students turn in their I/O planning, their logic design, as well as other preparation work, at least one week before the programming assignment is due and the programming assignment-related quiz is taken. The following is the point distribution for a typical 100 point project:

Logic design (at least one week earlier)	20 points
Programming assignment	40 points
Programming assignment-related quiz (on the due date of the assignment)	40 points
	100 points

The following is the scheme we use for calculating the final semester grade. It is based upon a total of 1000 points.

1.	Five programming projects	
	(50 percent of semester grade)	500 points
	Logical design (or walkthrough), 20 points each	
	Programming assignments, 40 points each	
	Programming assignment-related quiz, 40 points each	
2.	Three exams	
	Exam I	120 points
	Exam II	130 points
	Exam III	150 points
3.	Miscellaneous	
	(exercises, regular quizzes, etc.)	100 points
		1000 points

Some adjustments to the above schedule might be necessary, depending upon the circumstances of the course. This grade distribution system automatically prevents plagiarism of programming assignments.

VI. Conclusions and Summary.

Cheating on programming assignments is a problem of significant proportions, and it is one with which computing professionals and educators are rightly concerned. In an effort to stem the tide of this type of plagiarism, instructors have devised a wide range of methods for both detecting and preventing cheating. In this paper we have presented several methods in common use today and have contrasted them with two experimental methods which we feel respond to the problem in a more positive, effective manner. It is our contention that positive prevention using a simple, well-defined grading system is far more productive than the more negative suspicion-motivated approaches. From our own experience we can report that our students, both the strong ones and the weaker ones welcome this method, feeling above all that it is basically fair. This method is not perfect, particularly if the instructor is careless in designing the programming assignments and the associated quizzes, but when these are structured properly, we are assured that the students are learning the material more effectively and that the grades they receive accurately reflect their level of understanding. We, therefore, enthusiastically recommend these two methods, especially method Y.

We experimented only two months with method X. During that time we observed that some students (including the stronger ones) did well on the programming assignments, but poorly on the programming assignment-related quizzes. Assuming that the stronger students did indeed understand the material, we concluded that the strong students' poor performances, at least, were due to the students' having had a bad day or the programming assignment-related quiz being unable to represent the programming assignment. We concluded further, that these students in particular, and all the other students in general, were being inadvertently penalized too severely. We, therefore, developed method Y.

Our immediate experience with method Y was so successful that we in effect adopted the method right away. We now have had about a year's experience with it. At the end of a year we asked 71 students randomly for their personal reactions to the system. We correlated their responses with their grades which are summarized in the following table:

	A	B	C	D	F	
Strongly favor	5	5	4	2	0	16
Favor	11	15	11	8	4	49
Neutral	0	1	0	0	0	1
Disfavor	0	0	0	3	2	5
Strongly disfavor	0	0	0	0	0	0
	16	21	15	13	6	71

From the table it can be seen that those who favored the system most strongly had received the highest grades, and, conversely, those who disfavored the system most strongly had received the lowest marks.

An additional benefit from method Y is by having everyone take the quiz at the same time. Students are discouraged from procrastinating with their programming assignments. When everyone turns their work in on time, the chances are improved that their work will be evaluated more uniformly.

Finally, after a year's experience with method Y we can say that there is virtually no more plagiarism on programming assignments in our programming courses! This is a strong statement, we know, but private conversations with students, along with all other objective measuring methods at our disposal support this fact. Students respond overwhelmingly and affirmatively that it is a fair and equitable system.

VI. References

(1) Sam Grier, "A Tool that Detects Plagiarism in Pascal Programs", ACM SIGCSE Bulletin, Vol. 13, No. 1, February 1981.

(2) K. J. Ottenstein, "An Algorithmic Approach to the Detection and Prevention of Plagiarism", ACM SIGCSE Bulletin, Vol. 8, No. 4, Dec. 1976.

(3) John L. Donaldson, et al, "A Plagiarism Detection System", ACM SIGCSE Bulletin, Vol. 13, No. 1, February 1981.

(4) M. Shaw, et al, "Cheating Policy in a Computer Science Department", ACM SIGCSE Bulletin, Vol. 12, No. 2, July 1980.

(5) William Dodrill, et al, "Plagiarism in Computer Sciences Courses", ACM SIGCSE Bulletin, Vol. 13, No. 1, February 1981.

(6) S. Robinson, et al, "An Instructional Aid for Student Programs", ACM SIGCSE Bulletin, Vol. 12, No. 1, February 1980.