A Web-based System for Supporting Feedback-Enabled Teaching and Learning of Computer Programming*

S. Lam, L. Yuen, M. Y. Choy, C. K. Poon, F. L. Wang# and Y. T. Yu

Department of Computer Science
City University of Hong Kong, Hong Kong

Abstract. Assessment of computer programming exercises and assignments has created significant difficulties. We have developed a Web-based system called Programming Assignment aSsessment System (PASS) for automating the testing of students' programs. Students can submit their programs to the system for testing. Students will then receive instant feedback on the result of the executions. PASS has been used in programming courses since 2003, and has received favourable responses from both students and teachers. It has recently evolved to its third version with substantial upgrade in its overall performance and usability, together with major enhancements of functionalities. This paper describes the functions of PASS, demonstrating how it enables prompt and educationally useful feedback that supports the teaching and learning of computer programming.

1. Introduction

Assessment of programming exercises and assignments by manual inspection of code on paper is known to be tiring, inefficient and error-prone. Alternatively, the teacher may require students to perform demonstrations or to submit their program source code so that the teacher may execute the programs one by one. The latter methods allow a more reliable evaluation of the correctness of students' programs, but still they are labour-intensive and time-consuming. More importantly, such a practice is educationally undesirable because students generally have to wait for a long time before they can receive feedback from the teachers to help them improve their work.

Many different ways have been proposed and used to address this and similar problems that form barriers to the learning of computer programming. For details, the reader may refer to some recent surveys in the literature (e.g., Ala-Mutka, 2005; Gómez-Albarrán, 2005; Kelleher & Pausch, 2005). We have developed a Web-based system called Programming Assignment aSsessment System (PASS) for automating the testing of students' programs (Chong & Choy, 2004; Choy et al., 2005; Yu et al., 2006). Students can now submit their programs directly to the PASS, which executes their code automatically with test cases that are pre-defined by the teacher. Students will then receive instant feedback on the result of the executions. If the program fails, PASS can pinpoint exactly the position at which the program's actual output in the test run differs from the correct output according to the program's specification. In addition, the teacher can provide relevant comments to students by annotating each test case with its intended purpose. This kind of advice can be useful for students to understand in which scenarios their programs work incorrectly, thus indirectly giving hints and directions to desirable improvements.

* This work is supported in part by a Teaching Development Grant (project number: 6000118) from City University of Hong Kong.

# Corresponding author. Phone: 2788-7341. Fax: 2788-8614. Email: flwang@cityu.edu.hk

PASS has been used in programming courses since 2003, and has received favourable responses from both students and teachers (Chong & Choy, 2004; Choy et al., 2005; Yu et al., 2006). It has recently evolved to its third version with substantial upgrade in its overall performance and usability, together with major enhancements of functionalities to provide a greater variety of educationally useful feedback that facilitates the teaching and learning process. Detailed descriptions of previous versions of this system can be found in (Chong, & Choy, 2004; Choy et al., 2005; Yu et al., 2006).

Specifically, PASS now preserves all students' submissions and can analyze the information to facilitate the monitoring of the progress and performance of the whole class, selected groups of students, or each individual student. The computed statistics can be grouped by students to help identify those students with outstanding performance, as well as those who consistently lag behind. On the other hand, by grouping the submission results by individual programming problems, the Teacher can easily discover the common mistakes made or difficulties encountered by students. The Teacher may further inspect finer details of the problems by checking the effectiveness of each test case in detecting students' program faults. Temporal statistics are also available to view the usage and activity profiles of the students. All statistics are displayed in easy-to-read charts. To promote students' interests in learning, selected statistics can also be disclosed to students so that they themselves can better evaluate their own performance individually, as well as relatively to their peers. With this handy pedagogical tool, teachers are seeking to experiment new ways of constructing programming exercises for continuous improvement of their teaching and learning.

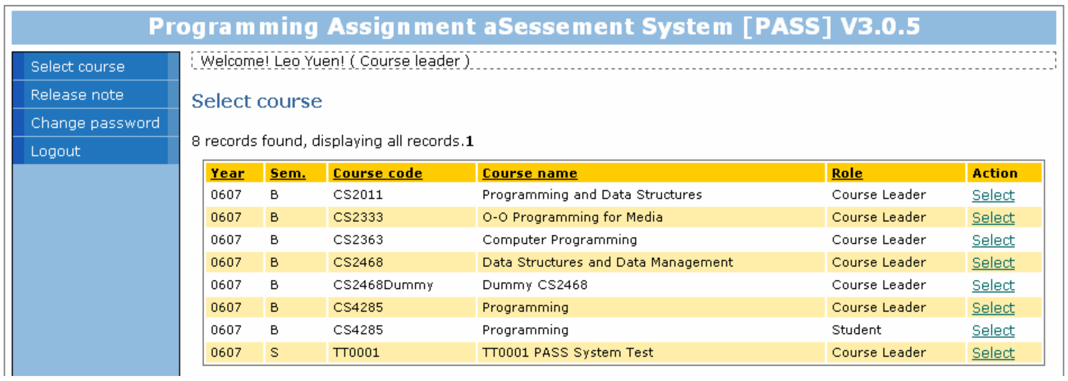
This paper will demonstrate the functions of the PASS. We will present a number of scenarios to show the user interaction with the system. PASS has defined five different roles of users in the system, namely, administrator, student, instructor, course developer, and course leader. They have different levels of privilege in the system:

1. Administrator is responsible for system maintenance and administrations.
2. Students will submit the program to the system for testing or grading.
3. Instructor is responsible for monitoring students' performance and assignment assessment.
4. Course developer is responsible for managing the programming problems.
5. Course leader has the privileges of both instructor's and course developer's, plus some extra course management functions.

To simplify our discussion, we will limit our presentation to the scenarios in which students and course leaders are involved. We will refer to a course leader as a 'teacher' in the rest of the paper. For each scenario, we will demonstrate various functions of the PASS by showing the associated detailed operations.

2. Functions of PASS

When a user logs on to PASS, a list of courses and roles will be displayed to the user, if the user has more than one role or has been enrolled to more than one course (Figure 1). The user can then select the course or role to log on to the system.


Programming Assignment aSessement System [PASS] V3.0.5

Welcome! Leo Yuen! (Course leader)

Select course

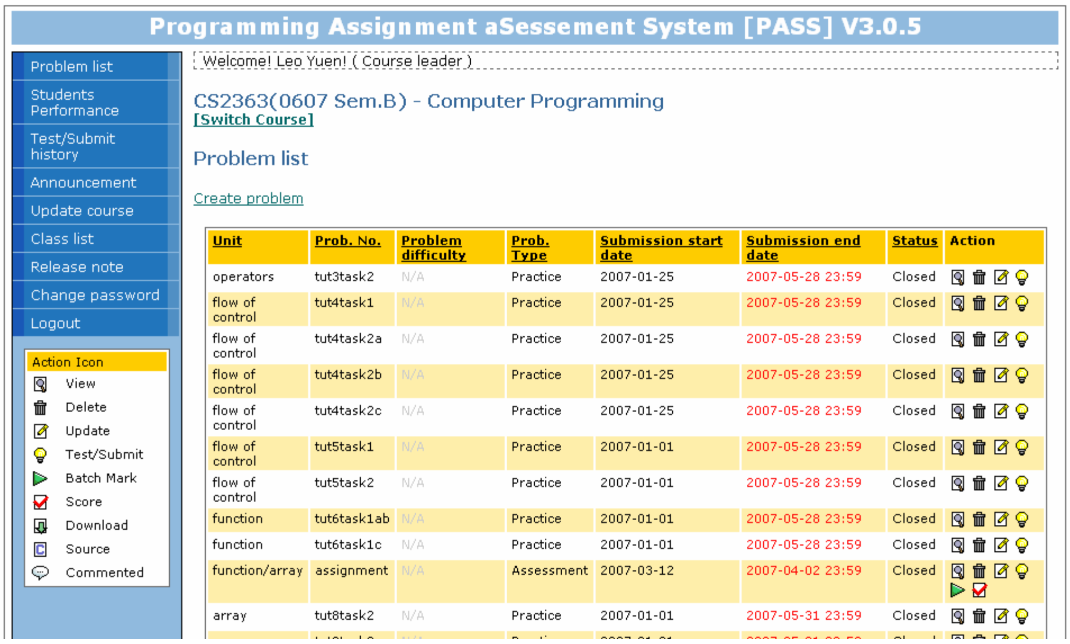
8 records found, displaying all records.**1**

Year	Sem.	Course code	Course name	Role	Action
0607	B	CS2011	Programming and Data Structures	Course Leader	Select
0607	B	CS2333	O-O Programming for Media	Course Leader	Select
0607	B	CS2363	Computer Programming	Course Leader	Select
0607	B	CS2468	Data Structures and Data Management	Course Leader	Select
0607	B	CS2468Dummy	Dummy CS2468	Course Leader	Select
0607	B	CS4285	Programming	Course Leader	Select
0607	B	CS4285	Programming	Student	Select
0607	S	TT0001	TT0001 PASS System Test	Course Leader	Select

Figure 1 Selection of courses

Problem management

After choosing the role and course (or if the user has only one role and is enrolled in only one course), a list of programming problems will be displayed. It lists all the active problems in the course (Figure 2). A student can view the details of an active problem and solve it, while a teacher has the right to manage the problems, such as creating, updating and deleting.


Programming Assignment aSessement System [PASS] V3.0.5

Action Icon	
🔍	View
🗑	Delete
📝	Update
💡	Test/Submit
▶	Batch Mark
☑	Score
⬇	Download
C	Source
💬	Commented

Welcome! Leo Yuen! (Course leader)

CS2363(0607 Sem.B) - Computer Programming
[Switch Course]

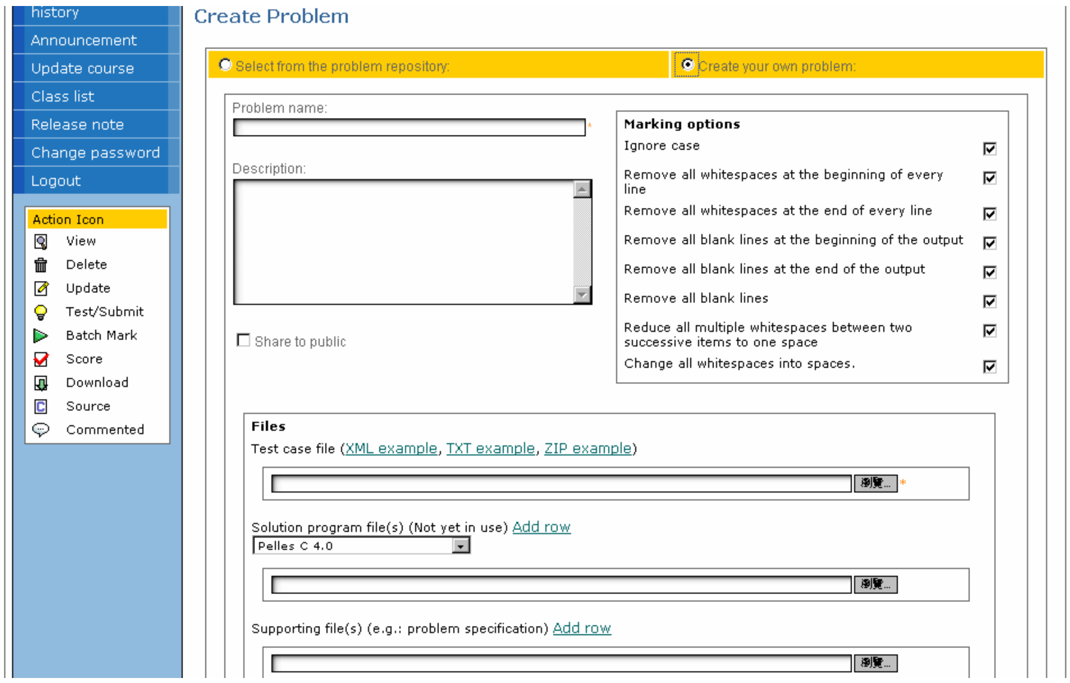
Problem list

Create problem

Unit	Prob. No.	Problem difficulty	Prob. Type	Submission start date	Submission end date	Status	Action
operators	tut3task2	N/A	Practice	2007-01-25	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
flow of control	tut4task1	N/A	Practice	2007-01-25	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
flow of control	tut4task2a	N/A	Practice	2007-01-25	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
flow of control	tut4task2b	N/A	Practice	2007-01-25	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
flow of control	tut4task2c	N/A	Practice	2007-01-25	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
flow of control	tut5task1	N/A	Practice	2007-01-01	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
flow of control	tut5task2	N/A	Practice	2007-01-01	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
function	tut6task1ab	N/A	Practice	2007-01-01	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
function	tut6task1c	N/A	Practice	2007-01-01	2007-05-28 23:59	Closed	🔍 🗑 📝 💡
function/array	assignment	N/A	Assessment	2007-03-12	2007-04-02 23:59	Closed	🔍 🗑 📝 💡 ▶ ☑
array	tut8task2	N/A	Practice	2007-01-01	2007-05-31 23:59	Closed	🔍 🗑 📝 💡

Figure 2 List of programming problems for students to solve

To create a problem (Figure 3), a teacher needs to prepare test cases for the problem that specify the example inputs and expected outputs. The teacher can provide the test cases in various formats, such as XML, zip or text file. Besides the compulsory test cases file, teacher can also provide supporting files such as problem specification documents or other files for the users to download. The teacher can also provide 'component files' that will be used during compilation or execution.



Create Problem

Select from the problem repository: [✓ Create your own problem:]

Problem name: *

Description:

☐ Share to public

Marking options

Ignore case	☑
Remove all whitespaces at the beginning of every line	☑
Remove all whitespaces at the end of every line	☑
Remove all blank lines at the beginning of the output	☑
Remove all blank lines at the end of the output	☑
Remove all blank lines	☑
Reduce all multiple whitespaces between two successive items to one space	☑
Change all whitespaces into spaces.	☑

Files

Test case file (XML example, TXT example, ZIP example) *

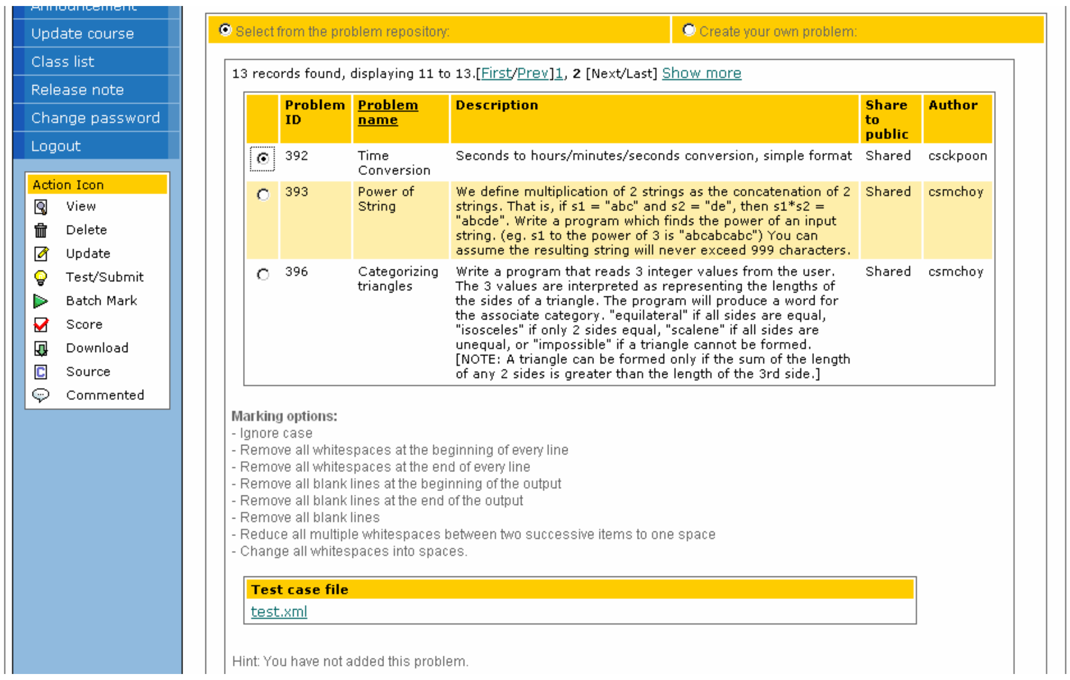
Solution program file(s) (Not yet in use) Add row

Pelles C 4.0

Supporting file(s) (e.g.: problem specification) Add row

Figure 3 Creation of a programming problem

The teacher can also specify how strict the actual outputs are to be checked against the expected outputs by choosing some ‘marking options’. For example, the teacher can give an option of ignoring all empty lines in the output, or specifying the checking to be case insensitive.



[✓ Select from the problem repository:] Create your own problem:

13 records found, displaying 11 to 13.[First/Prev]1, **2** [Next/Last] Show more

	Problem ID	Problem name	Description	Share to public	Author
●	392	Time Conversion	Seconds to hours/minutes/seconds conversion, simple format	Shared	csckpoon
○	393	Power of String	We define multiplication of 2 strings as the concatenation of 2 strings. That is, if s1 = "abc" and s2 = "de", then s1*s2 = "abcde". Write a program which finds the power of an input string. (eg. s1 to the power of 3 is "abcabcabc") You can assume the resulting string will never exceed 999 characters.	Shared	csmchoy
○	396	Categorizing triangles	Write a program that reads 3 integer values from the user. The 3 values are interpreted as representing the lengths of the sides of a triangle. The program will produce a word for the associate category. "equilateral" if all sides are equal, "isosceles" if only 2 sides equal, "scalene" if all sides are unequal, or "impossible" if a triangle cannot be formed. [NOTE: A triangle can be formed only if the sum of the length of any 2 sides is greater than the length of the 3rd side.]	Shared	csmchoy

Marking options:
- Ignore case
- Remove all whitespaces at the beginning of every line
- Remove all whitespaces at the end of every line
- Remove all blank lines at the beginning of the output
- Remove all blank lines at the end of the output
- Remove all blank lines
- Reduce all multiple whitespaces between two successive items to one space
- Change all whitespaces into spaces.

Test case file

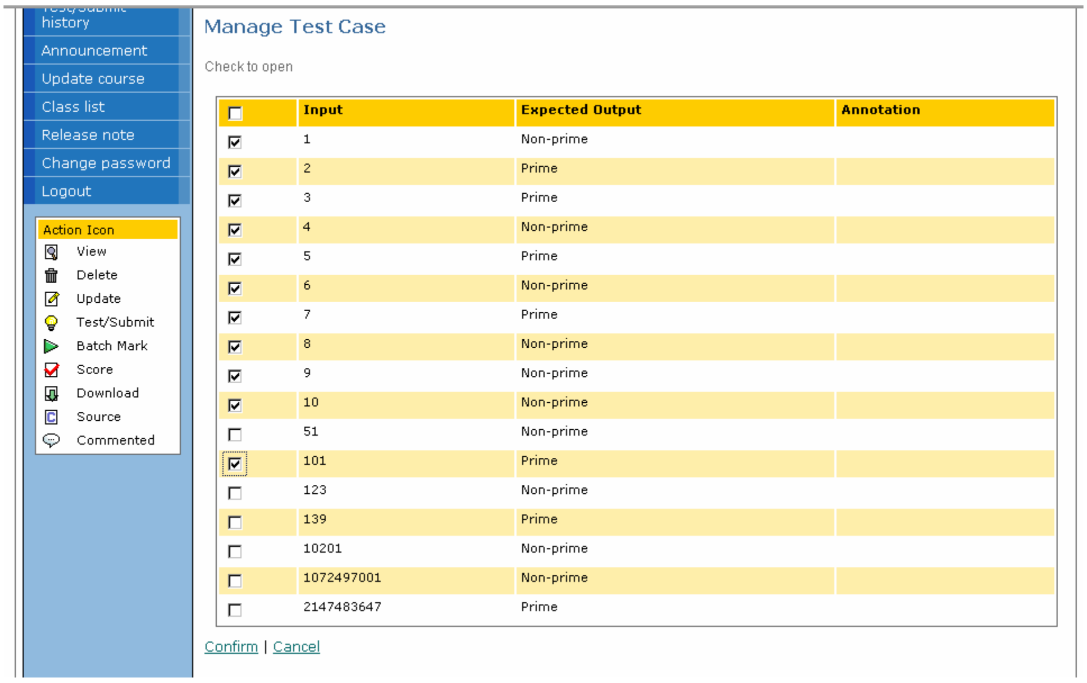
test.xml

Hint: You have not added this problem.

Figure 4 Problem repository

Other than creating a new problem from scratch, a teacher can choose to reuse problems stored in the ‘problem repository’, in which PASS stores all problems created and shared by other teachers (Figure 4). In the problem repository, a teacher can see those problems created by himself/herself, or those shared by other teachers.

After giving the information for a newly created problem (or choosing a problem from the repository), a list of test cases to this problem will be shown (Figure 5). The teacher may select which test cases, known as ‘open test cases’, to be executed when students test their programs on this problem. Test cases that are not selected are referred to ‘hidden test cases’, which will not be executed when a student tests their program on this problem. The ‘hidden test cases’ will be used by the system only when the teacher marks the programs submitted by the students.



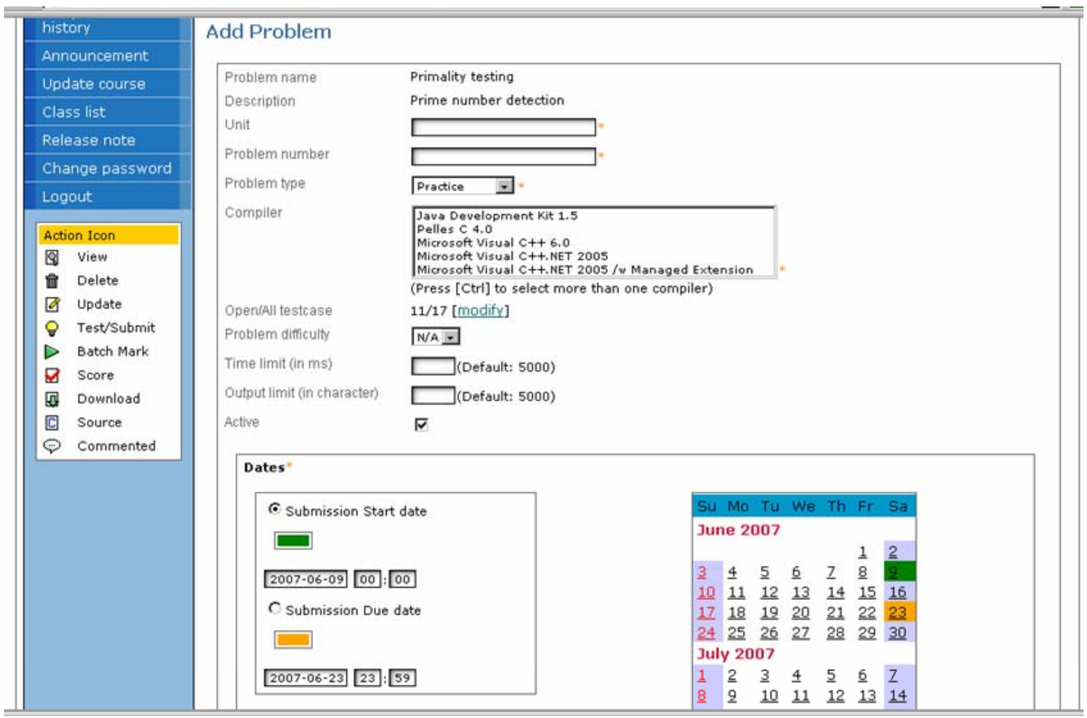
Manage Test Case

Check to open

☐	Input	Expected Output	Annotation
☑	1	Non-prime	
☑	2	Prime	
☑	3	Prime	
☑	4	Non-prime	
☑	5	Prime	
☑	6	Non-prime	
☑	7	Prime	
☑	8	Non-prime	
☑	9	Non-prime	
☑	10	Non-prime	
☐	51	Non-prime	
☑	101	Prime	
☐	123	Non-prime	
☐	139	Prime	
☐	10201	Non-prime	
☐	1072497001	Non-prime	
☐	2147483647	Prime	

Confirm | Cancel

Figure 5 Selection of test cases to be used for testing students’ programs



Add Problem

Problem name	Primality testing
Description	Prime number detection
Unit	*
Problem number	*
Problem type	Practice *
Compiler	Java Development Kit 1.5 Pelles C 4.0 Microsoft Visual C++ 6.0 Microsoft Visual C++.NET 2005 Microsoft Visual C++.NET 2005 /w Managed Extension * (Press [Ctrl] to select more than one compiler)
Open/All testcase	11/17 [modify]
Problem difficulty	N/A
Time limit (in ms)	(Default: 5000)
Output limit (in character)	(Default: 5000)
Active	☑

Dates*

● Submission Start date

🟩

2007-06-09 00 : 00

○ Submission Due date

🟧

2007-06-23 23 : 59

Su	Mo	Tu	We	Th	Fr	Sa
June 2007						
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
July 2007						
1	2	3	4	5	6	7
8	9	10	11	12	13	14

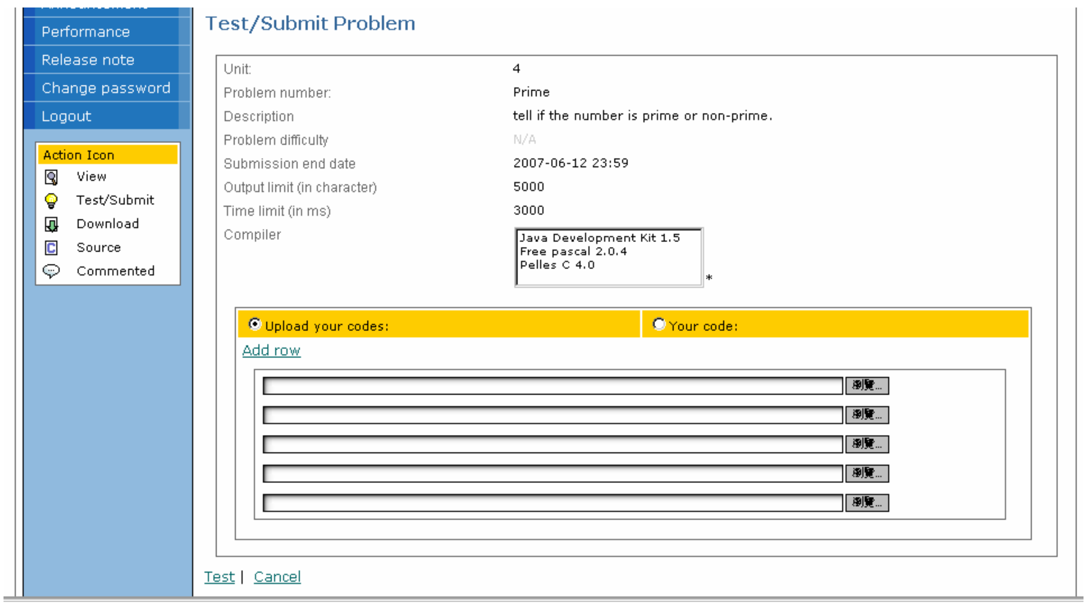
Figure 6 Adding a programming problem to a course

The teacher can further specify the information on how this problem is used in this course (Figure 6), such as the period of time during which this problem is visible to students, and the period of time during which they can submit their programs. Teacher can also define some settings on how long a program should be allowed for execution on each test case, and the maximum number of character outputs allowed.

Another important setting is the problem type, which can be either practice or assessment. The major difference is that an assessment type problem allows the students to submit their programs, while a practice type problem allows the students to test program only. When a problem has been added to a course, a new problem will be visible, and students can view and try to solve it, by submitting their programs.

Program submission

To solve a problem, a user needs to provide a program source code by uploading their files or typing into the input text area directly in the page. Besides, he/she may need to specify the compiler to use for compiling their programs, if the teacher has given them more than one choice of allowable compilers (Figure 7).



Test/Submit Problem

Unit:	4
Problem number:	Prime
Description	tell if the number is prime or non-prime.
Problem difficulty	N/A
Submission end date	2007-06-12 23:59
Output limit (in character)	5000
Time limit (in ms)	3000
Compiler	Java Development Kit 1.5 Free pascal 2.0.4 Pelles C 4.0 *

[✓ Upload your codes:] Your code:

Add row

Test | Cancel

Figure 7 Test or submit a program for a problem

Student can then choose to “test” or “submit” the program. After a student has chosen to “test” his program, the source code provided will be compiled and executed against each ‘open test case’ by feeding in the test inputs. The output given by the submitted program will be stored and compared to the expected output provided by the teacher.

The student can then view the execution result in the submission detail page. For each test case, the system shows the output given by the student’s program and the position at which the actual output starts to deviate from the expected output. All the error messages generated can be downloaded via the submission detail page whenever any error occurs during the process, such as when the source code fails to be compiled due to some syntax error.

If the user had chosen to “submit” the program, his/her program will only be compiled and not executed until the teacher has issued a batch run. This will be described later in the ‘Assignment assessment’ section.

Test/submit history

All the submitted files and execution results from any “test” or “submit” made to PASS will be stored, and can be retrieved in the “Test/submit history”.

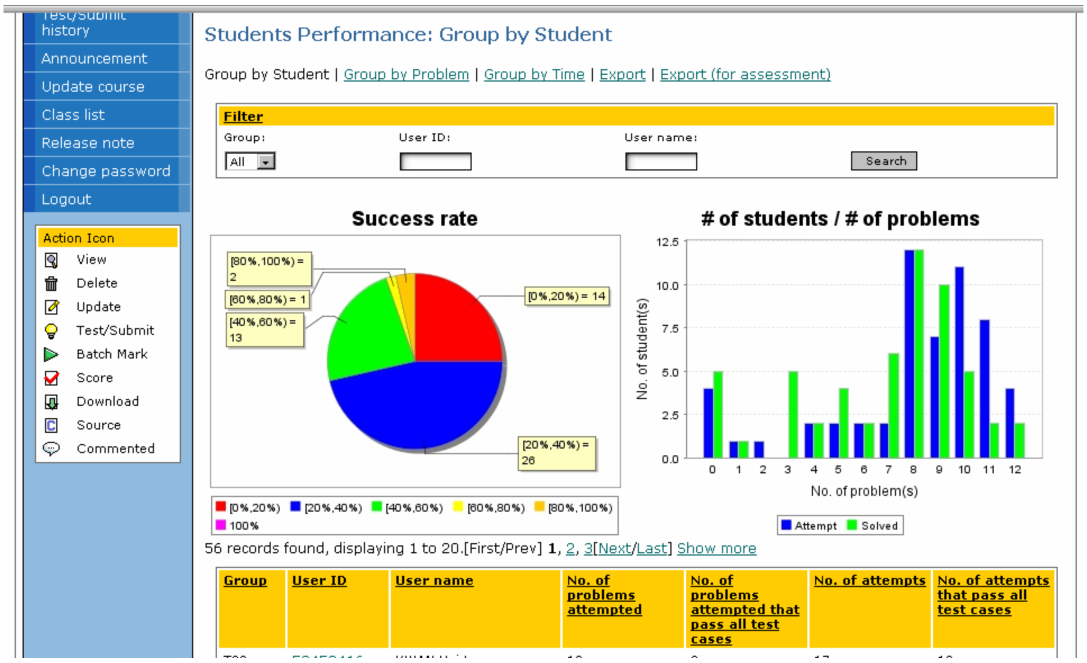
A student can find his/her own submission there and download any files that he/she had submitted before. This design allows the user to keep track of all the versions of programs he/she had created, and hence learn from when and what difficulties he/she had encountered in the past.

A teacher has the privilege to access all the test/submit history related to the course. This gives the teacher a very detailed log of the students’ activities in PASS. He/she can also give feedback to a student based on a specific submission. For example, if the teacher observes, from the test/submit history, that a student has attempted a problem many times repeatedly without success, he/she can download the submitted problem for inspection and provide some hint to the student.

Statistics

PASS also extracts useful aggregated information based on the raw data of test/submit history. The system computes various statistics to help the teacher monitor the performance of the class. The teacher can also make use of the filtering function to selectively display only the information related to the chosen criteria.

For example, when the statistics are grouped by student (Figure 8), the teacher can monitor the progress of each student to see how many problems they have attempted or solved. By using the filtering function, a teacher can compare the progress among different individuals or tutorial groups.



Students Performance: Group by Student

Group by Student | Group by Problem | Group by Time | Export | Export (for assessment)

Filter

Group: All

User ID:

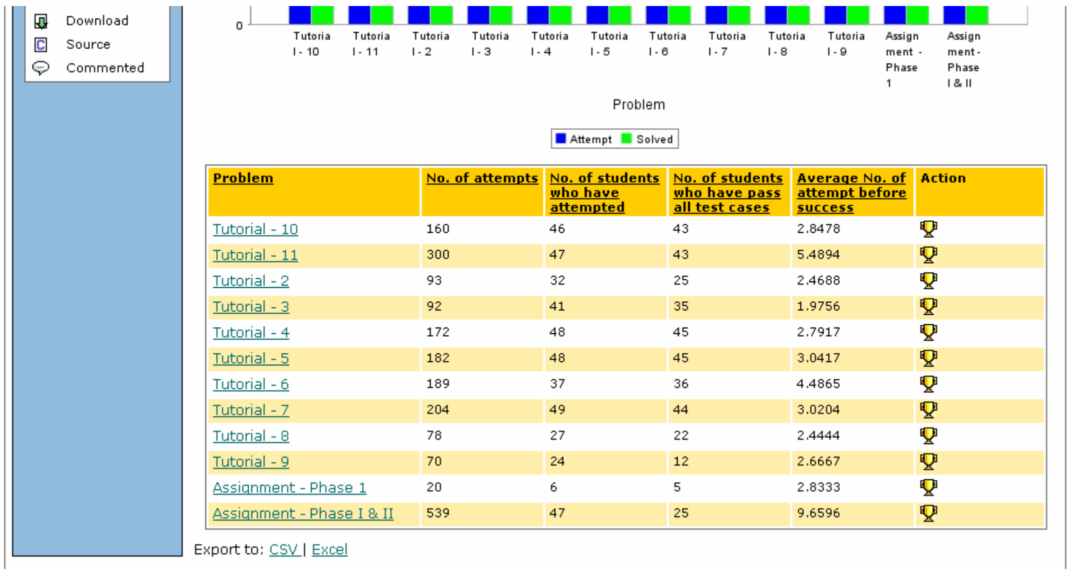
User name:

56 records found, displaying 1 to 20.[First/Prev] **1**, 2, 3[Next/Last] Show more

Group	User ID	User name	No. of problems attempted	No. of problems attempted that pass all test cases	No. of attempts	No. of attempts that pass all test cases

Figure 8 Statistics grouped by student

When the information is grouped by problem (Figure 9), the number of attempts by students and the number of students who have solved the problem are shown. The teacher can also see how many attempts are required, in average, before a problem is successfully solved by students.

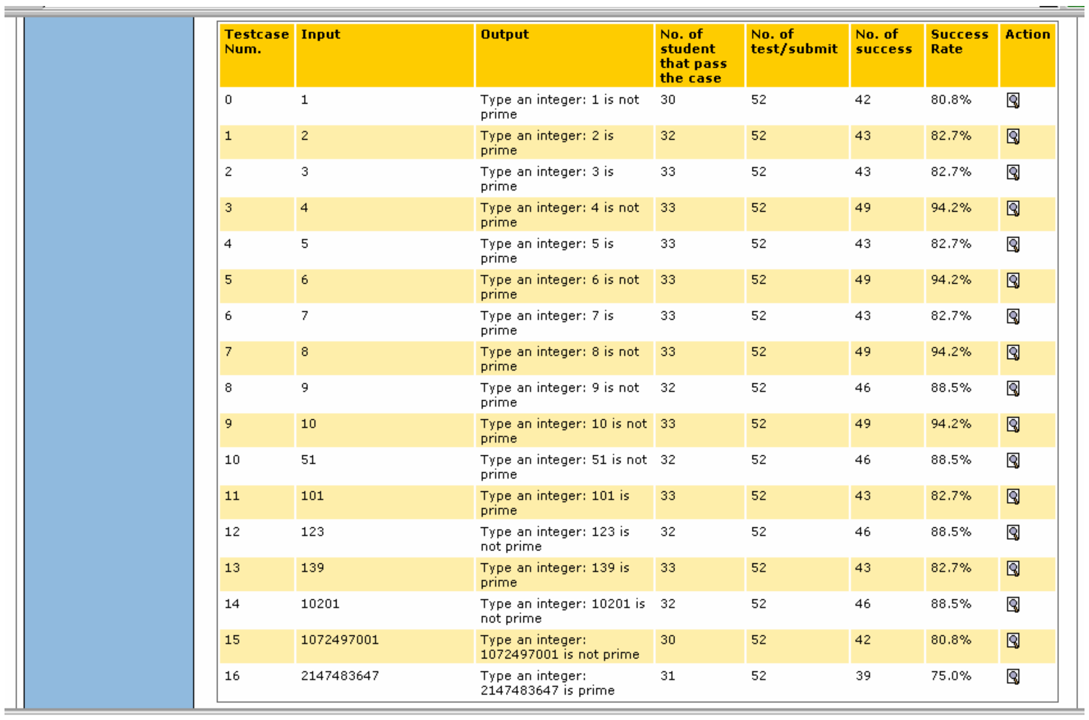


Problem	No. of attempts	No. of students who have attempted	No. of students who have pass all test cases	Average No. of attempt before success	Action
Tutorial - 10	160	46	43	2.8478	🏆
Tutorial - 11	300	47	43	5.4894	🏆
Tutorial - 2	93	32	25	2.4688	🏆
Tutorial - 3	92	41	35	1.9756	🏆
Tutorial - 4	172	48	45	2.7917	🏆
Tutorial - 5	182	48	45	3.0417	🏆
Tutorial - 6	189	37	36	4.4865	🏆
Tutorial - 7	204	49	44	3.0204	🏆
Tutorial - 8	78	27	22	2.4444	🏆
Tutorial - 9	70	24	12	2.6667	🏆
Assignment - Phase 1	20	6	5	2.8333	🏆
Assignment - Phase I & II	539	47	25	9.6596	🏆

Export to: CSV | Excel

Figure 9 Statistics grouped by problem

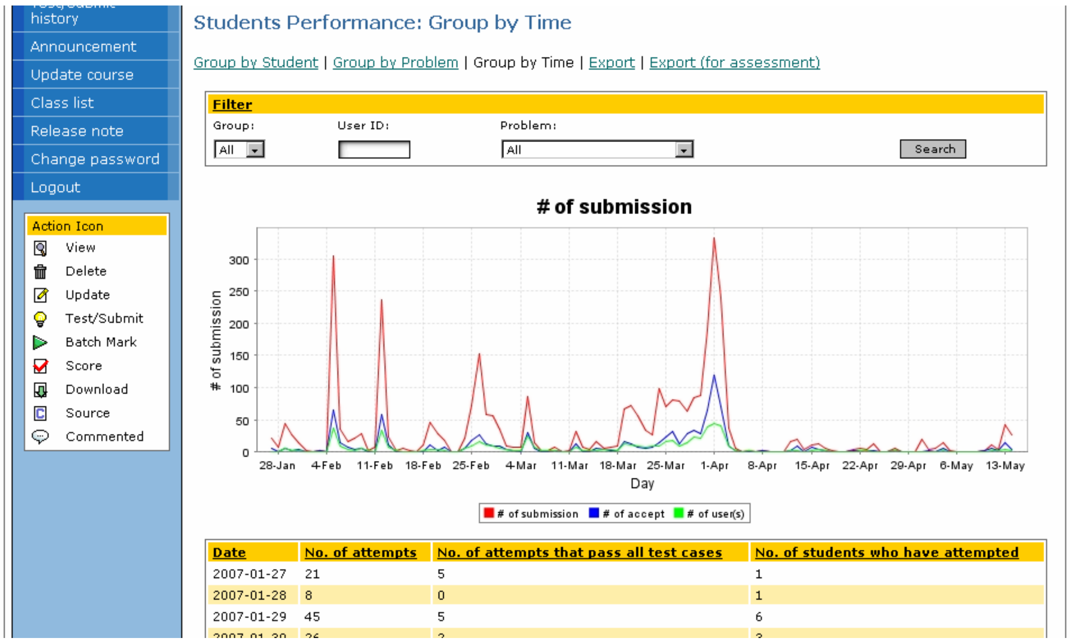
Besides, there is a rank list for each problem that shows all the students who have solved that problem, sorted by the time when they first solved it. This feature can be useful for hosting a competition. The teacher can also go into details of each problem to see the success rates among all the test cases (Figure 10). Further details such as how the students get incorrect answers and where they go wrong are also available.



Testcase Num.	Input	Output	No. of student that pass the case	No. of test/submit	No. of success	Success Rate	Action
0	1	Type an integer: 1 is not prime	30	52	42	80.8%	🔍
1	2	Type an integer: 2 is prime	32	52	43	82.7%	🔍
2	3	Type an integer: 3 is prime	33	52	43	82.7%	🔍
3	4	Type an integer: 4 is not prime	33	52	49	94.2%	🔍
4	5	Type an integer: 5 is prime	33	52	43	82.7%	🔍
5	6	Type an integer: 6 is not prime	33	52	49	94.2%	🔍
6	7	Type an integer: 7 is prime	33	52	43	82.7%	🔍
7	8	Type an integer: 8 is not prime	33	52	49	94.2%	🔍
8	9	Type an integer: 9 is not prime	32	52	46	88.5%	🔍
9	10	Type an integer: 10 is not prime	33	52	49	94.2%	🔍
10	51	Type an integer: 51 is not prime	32	52	46	88.5%	🔍
11	101	Type an integer: 101 is prime	33	52	43	82.7%	🔍
12	123	Type an integer: 123 is not prime	32	52	46	88.5%	🔍
13	139	Type an integer: 139 is prime	33	52	43	82.7%	🔍
14	10201	Type an integer: 10201 is not prime	32	52	46	88.5%	🔍
15	1072497001	Type an integer: 1072497001 is not prime	30	52	42	80.8%	🔍
16	2147483647	Type an integer: 2147483647 is prime	31	52	39	75.0%	🔍

Figure 10 Statistics grouped by test case

When the statistics are grouped by time (Figure 11), the teacher can observe the usage of PASS, deduce which class of students is lagging behind, and adjust the pace of the teaching. It gives information on students' practice behavior, such as which student group is working harder (that is, practices more frequently).



Students Performance: Group by Time

Date	No. of attempts	No. of attempts that pass all test cases	No. of students who have attempted
2007-01-27	21	5	1
2007-01-28	8	0	1
2007-01-29	45	5	6

Figure 11 Statistics grouped by time

To allow students to keep track of their own progress, selected statistics can also be released to students. They can then compare their own performance to that of the class average. All statistics are accompanied with graphical presentation so that important information can be figured out at a glance. Besides, all statistics can be exported in popular spreadsheet formats for archive purposes or further analysis.

Assignment assessment

For assessment problems, the teacher can start a batch run on all the programs submitted by the students against all the test cases kept in PASS, including those hidden test cases. Upon completion of the batch run, the teacher can get a result list, showing the number of the success test cases and hence the score, for each student.

To detect plagiarism, PASS provides a function to check the similarity among the submitted source files. For better management, PASS can also group the students whose similarity scores are higher than a given threshold. This helps the teacher to investigate and identify who are copying from the same source.

The current plagiarism detection algorithm calculates the similarity score based on the notion of longest common subsequences. It can easily be replaced by other more sophisticated similarity checking algorithms by implementing the specific programming interface provided by the PASS.

3. Conclusion

This paper has described the functionalities of the latest version of PASS. The substantially enhanced system has been used for a semester and has received very encouraging feedback from the users. Future enhancements include facilities to allow better problem management, such as test case editor and customized marker editor, in order to save time of the teacher.

Reference

Ala-Mutka, K.M. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education*, *15*(2), 83-102.

Chong, S.L., & Choy, M. (2004). Towards a Progressive Learning Environment for Programming Courses. In *Proceedings of the 4th International Conference on Web-based Learning (ICWL 2004),* (pp. 200-205).

Choy, M., Nazir, U., Poon, C.K., & Yu, Y.T. (2005). Experiences in using an automated system for improving students learning of computer programming. In *Proceedings of the 4th International Conference on Web-based Learning (ICWL 2005),* (pp. 267-272).

Gómez-Albarrán, M. (2005). The teaching and learning of programming: A survey of support software tools. *The Computer Journal*, *48*(2), 130-144.

Kelleher, C. & Pausch R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, *37*(2), 83-137.

Yu, Y.T., Poon, C.K., & Choy, M. (2006). Experiences with the PASS: Developing and Using a Programming Assignment aSsessment System. In *Proceedings of the 6th International Conference on Quality Software (QSIC 2006),* (pp. 360-368). IEEE Computer Society Press.

Biography

Mr S. Lam is the system designer and constructer of PASS (version 3). He is a graduate of BSc (Hons) Computer Science from City University of Hong Kong in the year 2006. He is responsible for the design of the architecture and functions and is also one of the major developers of PASS (version 3).

Mr Leo Yuen received the MPhil and BSc degrees in Computer Studies from City University of Hong Kong. His research interests include information retrieval and XML query processing.

Dr Y. T. Yu is an associate professor at the Department of Computer Science, City University of Hong Kong. His research interests include software engineering, software testing and quality, and computers in education. He has published widely in these areas, and has also authored two textbooks for Computer Studies. He is on the editorial board of the International Journal of Web Engineering and Technology.

Dr C. K. Poon is an associate professor at the Department of Computer Science, City University of Hong Kong. His research interests include theoretical computer science, database, information retrieval and computers in education.

Dr F. L. Wang is a lecturer at the Department of Computer Science, City University of Hong Kong. His research interests include automatic document summarization, information retrieval, information systems, and computers in education. He has published articles in leading journals and proceedings of international conferences in these areas. He has also co-authored two books in e-learning.

Dr M. Y. Choy is an instructor at the Department of Computer Science, City University of Hong Kong. Her research interests include computers in education, assistive technology, and image processing. She is experienced in teaching programming courses to non-Computer Science students, and is currently working on publishing a workbook to assist the learning of computer programming.