Education and Information Technologies (2019) 24:2779–2796
https://doi.org/10.1007/s10639-019-09896-1

Learning problem solving skills: Comparison of E-learning and M-learning in an introductory programming course

Sohail Iqbal Malik[1] · Roy Mathew[1] · Rim Al-Nuaimi[1] · Abir Al-Sideiri[1,2] · Jo Coldwell-Neilson[3]

Received: 13 January 2019 / Accepted: 7 March 2019 / Published online: 13 March 2019

Abstract
Problem solving skills are considered an integral part of grasping the precise concepts of the programming domain for novices in introductory programming (IP) courses. But these skills are mostly covered only in early lectures of such courses or are included in just a few early chapters of some relevant textbooks. Consequently, high failure and dropout rates are often reported in IP courses. In this study, we developed and introduced an application, called PROBSOL, which is designed to focus on, and enhance, novice programmers' problem solving skills. Two versions of the application were developed, a web-based version and a mobile app. The applications were based on pseudo-code techniques. A survey was implemented to collect students' feedback and semi-structured interviews were conducted to collect instructors' opinion about the applications. Moreover, final exam grades over two semesters were compared to determine the impact of including the PROBSOL applications in the course. The results show that both students and instructors appreciated the applications and that their use supports students' cognitive gains and engagement. Moreover, they promote students' affective engagement in the IP course. The use of the applications improves novices' programming understanding, logic capabilities and problem solving skills. However, students preferred the mobile version rather than the web-based version during the course. The comparison of students' grades showed that the students' achievements were improved and attrition rates were reduced after introducing the PROBSOL applications in the course.

Keywords Problem solving skills · Computer programming · M-learning · Pseudo-code · E-learning

✉ Sohail Iqbal Malik
sohail_iqbal9@hotmail.com

Extended author information available on the last page of the article

1 Introduction

Learning to program is considered a difficult task for a substantial number of novice programmers (Reardon and Tangney 2014; Malik and Coldwell-Neilson 2018). Novices have to focus on a set of skills (problem solving, syntax and semantics) at the same time (Malik 2019). Further, introductory programming (IP) courses require students to learn the theoretical concepts of the programming domain and then practice these concepts in designing and coding the given problem statement. Consequently, this process challenges the cognitive processes of many novice programmers, pushing them into information overload (Shuhidan 2012).

Problem solving skills are an integral component in learning to program. However, some studies suggest that novice programmers spend more time in acquiring programming knowledge (syntax and semantics) rather than learning problem solving strategies (De Raadt 2008; Webster 1994; Kölling and Rosenberg 1996; Ala-Mutka 2004; Iqbal and Harsh 2013). Some textbooks used in IP courses cover problem solving strategies only in the early chapters (De Raadt 2005). Robins et al. (2003) pointed out that 'typical introductory programming textbooks devote most of their content to presenting knowledge about a particular language' (p. 141). Ala-Mutka (2004) described that learning to program comprises of several activities such as learning the programming language features, program design and comprehension. The traditional approach used in teaching IP courses starts with programming knowledge (syntax and semantics) of a particular language, paying less attention to problem solving strategies.

In this study, we developed and introduced the PROBSOL application to enhance novice programmers' problem solving skills. The application facilitated the learning process of students in two different modes of learning, web-based called E-PROBSOL, and a mobile app called M-PROBSOL. Both applications are based on pseudo-code techniques. The applications covered all the teaching topics of the course and were offered for the whole semester. Moreover, they provided opportunities for students to practice problem solving strategies in the course without having to worry about the syntax of the programming language.

This paper is organized as follows. The literature review is presented first, followed by a brief introduction to the overall design of PROBSOL applications. The methodology adopted for this study is then described, followed by a presentation of the research results and discussion. The paper concludes with the outcomes of the study, recommendations for practitioners and areas for further work.

2 Literature review

Problem solving skills are considered as an important and fundamental component in the programming domain (Malik et al. 2019). De Raadt (2008) provided a definition of problem solving as 'A mechanism for achieving a solution to a programming problem' (p. ix) which is also applicable in this context. Novice programmers should focus on problem solving strategies in an IP course. In contrast, researchers concluded that novice programmers focus more on coding instead of paying attention to problem solving skills as well. (Yoo et al. 2012; Iqbal and Harsh 2013). This finding is also reinforced in the IP textbooks. De Raadt et al. (2005) reviewed forty textbooks used in an introductory programming course. He concluded that six books out of forty integrated problem solving strategies throughout all topics. Most of the analyzed books focused more on the syntax and semantics of the programming languages. In some cases, problem solving skills were covered only in early chapters of the book. Yet as early as 1996, Winslow discussed that, it is important for novices to focus on programming language syntax, semantics and problem solving skills concurrently. The ACM-IEEE Joint Task Force on Computing Curricula (2013) emphasizes that novice programmers should equally focus on problem solving strategies and programming knowledge (syntax and semantics). Similarly the Australian Computer Society (2013) recommends a focus on problem solving strategies and programming knowledge in programming courses.

Research studies suggested different problem solving strategies for novice programmers. De Raadt (2008) introduced explicit instruction on programming strategies based on Goals and Plans as proposed by Soloway (1986). In this strategy, the instructor determines the goals that need to be achieved within a given problem and then these goals are mapped to plans. Winslow (1996) divided program problem solving into four steps:

1) Understand the problem
2) Determine how to solve the problem:

   a) in some form, and
   b) in computer compatible form
      (Note that generally, students have trouble going from step 2a to step 2b.)

3) Translate the solution into a computer language program, and
4) Test and debug the program' (p. 19)

Malik and Coldwell-Neilson (2016) proposed an ADRI (Approach, Deployment, Result, Improvement) approach to promote problem solving skills in the IP course. The first stage (Approach) uses two different problem solving strategies (pseudo code and flowchart) to solve the given problem statement. Moreover, an ADRI editor was prepared to promote the problem solving strategies in the course. Novices are encouraged to practice problem solving strategies throughout the course. Consequently, achievement of student learning outcomes was shown to improve with the ADRI approach compared to outcomes prior to its introduction. Moreover, IP course instructors also found that the ADRI approach enhances the problem solving skills of novice programmers (Malik and Coldwell-Neilson 2017a). Papadakis et al. (2016) introduced two block-based programming environments (scratch and App Inventor) in the teaching and learning process of introductory programming in secondary education. They concluded that both environments were attractive platforms for introducing fundamental concepts in computer programming. Kanaki and Kalogiannakis (2018) introduced digital environment PhysGramming to children in early childhood education to provide them familiarity with the basic principles of object-oriented programming and computational thinking.

Koulouri et al. (2015) also introduced problem-solving training for students in an IP course before starting programming. Students' performance across two groups was compared, those with problem solving training and those without. They found that students who had completed the problem solving training performed better than those students who did not.

Different tools have been introduced in IP courses to enhance problem solving skills. Oda et al. (2015) used statistical machine translation (SMT) framework which automatically generates pseudo-code from a source code of the program. Taheri et al. (2015) introduced a web based environment named progranimate (see http://www.progranimate.com/) which uses flowcharts to find a solution for a given problem statement for novice programmers. Malik (2016b) introduced the ADRI editor for novice programmers which uses a pseudo-code technique to solve given programming problems. Hooshyar et al. (2015) introduced a Flowchart-based Intelligent Tutoring System (FITS) for improving problem-solving skills of novice programmers.

3 Research questions

It is evident from the above discussion that problem solving skills are an important component for introductory programming courses. Novices should focus on and practice problem solving strategies throughout IP courses. This research project proposes the PROBSOL applications to support teaching and learning processes in IP courses to enhance the problem solving skills of novice programmers. Moreover, this research proposes three research questions to determine the impact of the PROBSOL applications on novice programmers. These are:

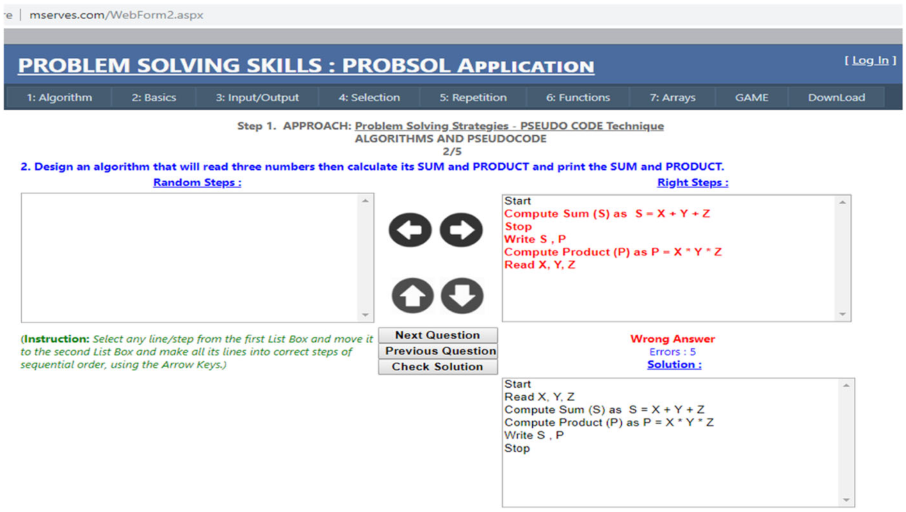
RQ1: What are the perceptions of students regarding the PROBSOL applications in the introductory programming course?
RQ2: What are the perceptions of instructors regarding the PROBSOL applications in the introductory programming course?
RQ3: What is the impact of the PROBSOL applications on the students learning in the introductory programming course?

4 Introduction to the PROBSOL applications

The PROBSOL application is based on pseudo-code techniques. It was offered in two different modes, a web-based version (E-PROBSOL) and a mobile app (M-PROBSOL). The M-PROBSOL app is available from the Google store. Features and functionalities of the M-PROBSOL are same as the E-PROBSOL application. Figure 1 shows the E-PROBSOL interface and Fig. 2 the M-PROBSOL interface respectively.

Both versions of the application include a comprehensive selection of topics that would be expected to be included in an IP course. Questions relating to each topic are grouped together as menu items. Navigation through the topics and questions are easily achieved by clicking at the required menu and its menu items respectively. Moreover, the user can change question by clicking Next and Previous Question buttons.

The application consists of three list boxes (Random Steps, Right Steps and Solution). The Random Steps list box shows all the Pseudo-code steps as part of the solution for the given problem statement. Students select the appropriate step from the Random Steps list box and then click the Right Arrow Button, so that the selected step will get moved to Right Step list box as shown in Fig. 3. The correct pseudo-code solution for the given problem is shown in the Solution list box as shown in Fig. 1.


mserves.com/WebForm2.aspx

PROBLEM SOLVING SKILLS : PROBSOL APPLICATION [Log In]

1: Algorithm | 2: Basics | 3: Input/Output | 4: Selection | 5: Repetition | 6: Functions | 7: Arrays | GAME | DownLoad

Step 1. APPROACH: Problem Solving Strategies - PSEUDO CODE Technique
ALGORITHMS AND PSEUDOCODE
2/5

2. Design an algorithm that will read three numbers then calculate its SUM and PRODUCT and print the SUM and PRODUCT.

Random Steps :

Right Steps :

```
Start
Compute Sum (S) as  S = X + Y + Z
Stop
Write S , P
Compute Product (P) as P = X * Y * Z
Read X, Y, Z
```

(Instruction: Select any line/step from the first List Box and move it to the second List Box and make all its lines into correct steps of sequential order, using the Arrow Keys.)

Next Question
Previous Question
Check Solution

Wrong Answer
Errors : 5
Solution :

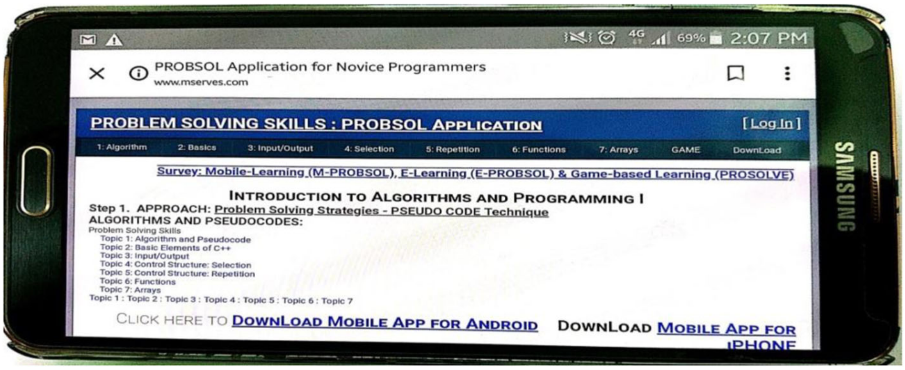
```
Start
Read X, Y, Z
Compute Sum (S) as  S = X + Y + Z
Compute Product (P) as P = X * Y * Z
Write S , P
Stop
```

Fig. 1 E-PROBSOL application

The 'Check Solution button' is activated only when the user moves all the pseudo-code steps for the given problem statement from Random Steps list box to Right Steps list box. The 'Check Solution button' compares the user solution with the correct solution shown in Solution list box. If the user solution is incorrect, the Right Steps list box displays the lines or steps containing errors in red as shown in Fig. 3. The application also counts the number of errors and displays it between Right steps and Solution list boxes. Further, the application also displays different warning and error messages.

5 Research methodology and design

The first research question was explored by collecting students' responses after introducing the PROBSOL applications in the teaching and learning process of the introductory programming (IP) course. Both versions of the application were offered to all


69% 2:07 PM

PROBSOL Application for Novice Programmers
www.mserves.com

PROBLEM SOLVING SKILLS : PROBSOL APPLICATION [Log In]

1: Algorithm | 2: Basics | 3: Input/Output | 4: Selection | 5: Repetition | 6: Functions | 7: Arrays | GAME | DownLoad

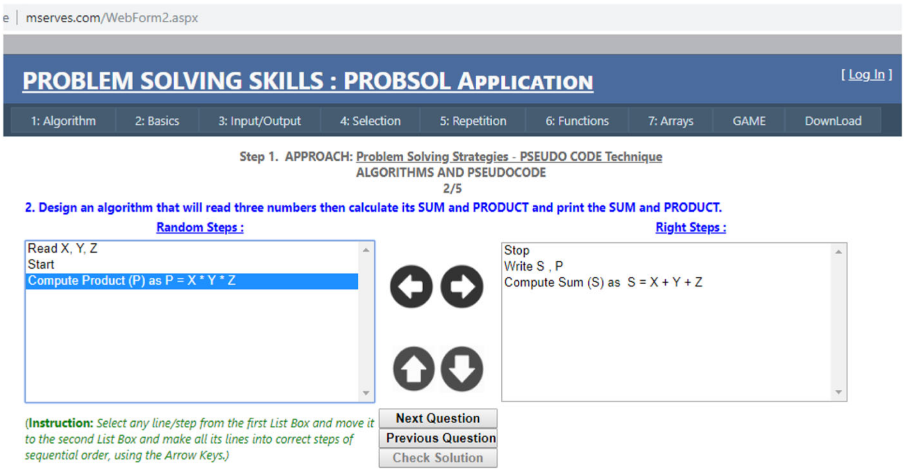
Survey: Mobile-Learning (M-PROBSOL), E-Learning (E-PROBSOL) & Game-based Learning (PROSOLVE)

INTRODUCTION TO ALGORITHMS AND PROGRAMMING I

Step 1. APPROACH: Problem Solving Strategies - PSEUDO CODE Technique
ALGORITHMS AND PSEUDOCODES:
Problem Solving Skills
Topic 1: Algorithm and Pseudocode
Topic 2: Basic Elements of C++
Topic 3: Input/Output
Topic 4: Control Structure: Selection
Topic 5: Control Structure: Repetition
Topic 6: Functions
Topic 7: Arrays
Topic 1 : Topic 2 : Topic 3 : Topic 4 : Topic 5 : Topic 6 : Topic 7

CLICK HERE TO DOWNLOAD MOBILE APP FOR ANDROID DOWNLOAD MOBILE APP FOR IPHONE

Fig. 2 M-PROBSOL application



Step 1. APPROACH: Problem Solving Strategies - PSEUDO CODE Technique
ALGORITHMS AND PSEUDOCODE
2/5

2. Design an algorithm that will read three numbers then calculate its SUM and PRODUCT and print the SUM and PRODUCT.

Random Steps:
```
Read X, Y, Z
Start
Compute Product (P) as P = X * Y * Z
```

Right Steps:
```
Stop
Write S , P
Compute Sum (S) as  S = X + Y + Z
```

(**Instruction:** *Select any line/step from the first List Box and move it to the second List Box and make all its lines into correct steps of sequential order, using the Arrow Keys.)*

Fig. 3 E-PROBSOL list boxes

the students registered for the IP course during the semester 1, 2018–19. Both applications were offered to the students for the whole semester. The survey was deployed to collect students' feedback. The second research question was investigated by conducting semi-structured interviews with all instructors of the IP course for the semester 1, 2018–19. The third research question was explored by comparing the final grades of students over two semesters to determine the impact of the PROBSOL applications on the students' achievements.

5.1 Population and ethical consideration

The survey was conducted with the IP students of Buraimi University College, Oman. 65 students participated in the survey. Of the 65 students, 52 (80%) were female and 13 were (20%) male. This trend is rare in IT so it provides us a unique opportunity to conduct this study where the majority of participants were female. The college offers degree programs in two shifts. The morning shift is only for female students and evening shift is offered for both male and female students. This particular system is not very common in Oman so our college attracts more female students compared to male students.

The semi-structured interviews were conducted with two instructors of the IP course. The data were collected after obtaining ethical approval from the college. Student participation was optional and voluntary in the survey. Data collection was anonymous in the survey. Interview data is presented without showing the instructors' identity. Students' grades were collected after a de-identification process was performed by the registration department.

5.2 Preparation of the survey

The survey was prepared to explore the effectiveness of the PROBSOL applications in enhancing the problem solving skills of novice programmers in the IP course. The survey consists of 35 closed-ended questions. The student survey consists of following three parts.

1. Demographic details
2. PROBSOL Application: Programming Concepts
3. E-PROBSOL and M-PROBSOL Applications

The demographic questions included students' major, degree, age, gender and prior programming experience. Fifteen statements related to the course content covered were included in the second part of the survey and a five point Likert scale was used for students to indicate their agreement with each statement (1 = strongly disagree to 5 = strongly agree). The third part of the survey focused on questions (again, 15 questions) comparing E-PROBSOL with M-PROBSOL with a five point Likert scale (1 = strongly disagree to 5 = strongly agree).

6 Results

This section describes the results of this study. The first research question was explored by conducting a survey with the students who finished the IP course after introducing the PROBSOL applications in the course. The second research question was investigated by conducting semi-structured interviews with instructors of the IP course. The third research question was explored by comparing the final grades of students over two semesters to determine the impact of the PROBSOL applications on the students' achievements.

The survey data was analyzed by using IBM SPSS (Statistical Package for Social Science, version 20) software. Two descriptive statistical methods (mean and frequency) were used to analyze the collected data.

6.1 Analysis of student survey

This section addresses the first research question which is:

> *RQ 1: What are the perceptions of students regarding the PROBSOL application in the introductory programming course?*

6.1.1 Demographic details

The demographic details of the respondents are as follows:

- 49.2% respondents are enrolled in the Information Systems major, compared to 26.2% from Computer Science and 24.6% from Software Engineering.
- Most of the respondents (55.4%) are pursuing their studies at bachelor level as compared to 7.7% for advanced diploma and 36.9% for diploma.
- 87.7% respondents are between 18 to 28 years old, compared to 7.7% under 18 years, and 4.6% from 45 years and over. There were no students between the ages of 29 and 44.

- The majority of the respondents (80%) were female compared to 20% male.
- A majority of the respondents (67.7%) do not have any prior programming experience.

6.1.2 PROBSOL application: Programming concepts

Table 1 shows the respondents' perceptions related to PROBSOL in learning the basic concepts of computer programming. Both the means and frequencies of the responses are included in the table.

Respondents perceived 'Understanding the programming concepts (4.03)', 'Design a program to solve a certain task (4.03)', 'Learn problem solving skill (pseudo code) (4.02)' as the most three significant areas in learning the basic concepts of computer programming. Respondents also found that PROBSOL helped them to: 'Learn the course content (3.90)', 'Focus only on problem solving skills (3.85)', 'Understanding programming structures (3.85)', 'Participate in the course activity in ways that enhanced my learning (3.68)' and 'Develop confidence in the subject area (3.66)'.

6.1.3 PROBSOL application: Concepts

This part of the survey covered all the main topics taught in the IP course. The means and frequencies of the responses are shown in Table 2.

Table 1 PROBSOL application: usability

PROBSOL application helped me to:	Mean	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Non response	Total
Understand the programming concepts	4.03	23.1%	56.9%	20.0%	0%	0%	0%	100%
		15	37	13	0	0	0	65
Learn problem solving skill (Pseudo code)	4.02	26.2%	53.8%	15.4%	4.6%	0%	0%	100%
		17	35	10	3	0	0	65
Focus only on problem solving skills	3.85	20%	52.3%	21.5%	4.6%	1.5%	0%	100%
		13	34	14	3	1	0	65
Understand programming structures	3.85	23.1%	41.5%	26.2%	3.1%	1.5%	4.6%	100%
		15	27	17	2	1	3	65
Learn the course contents	3.90	20.0%	58.5%	10.8%	4.6%	3.1%	3.1%	100%
		13	38	7	3	2	2	65
Design a program to solve a certain task	4.03	26.2%	53.8%	15.4%	1.5%	1.5%	1.5%	100%
		17	35	10	1	1	1	65
Participate in the course activity in ways that enhanced my learning	3.68	15.4%	44.6%	32.3%	7.7%	0%	0%	100%
		10	29	21	5	0	0	65
Develop skills that apply to academic career and/or professional life.	3.97	23.1%	55.4%	12.3%	4.6%	1.5%	3.1%	100%
		15	36	8	3	1	2	65
Develop confidence in the subject area	3.66	12.3%	52.3%	24.6%	6.2%	3.1%	1.5%	100%
		8	34	16	4	2	1	65

Table 2 PROBSOL application: concepts

PROBSOL helped me to learn:	Mean	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Non response	Total
Designing an algorithm	4.20	38.5%	44.6%	15.4%	1.5%	0%	0%	100%
		25	29	10	1	0	0	65
Input / Output	4.16	30.8%	56.9%	6.2%	4.6%	0%	1.5%	100%
		20	37	4	3	0	1	65
Selection structure (if statement)	3.90	24.6%	43.1%	26.2%	1.5%	1.5%	3.1%	100%
		16	28	17	1	1	2	65
Repetition structures (loops)	3.60	16.9%	33.8%	41.5%	7.7%	0%	0%	100%
		11	22	27	5	0	0	65
Functions	3.90	18.5%	50.8%	24.6%	1.5%	0%	4.6%	100%
		12	33	16	1	0	3	65
Arrays	3.66	16.9%	41.5%	35.4%	3.1%	3.1%	0%	100%
		11	27	23	2	2	0	65

The respondents perceived 'Designing an algorithm (4.20)', 'Input/output (4.16)', 'Selection structure (if statement) (3.90)' and 'Functions (3.90)' as the most significant learning areas of programming concepts after introducing the PROBSOL application in the IP course. The respondents also found learning gain in the programming concepts of 'Arrays (3.66)' and 'Repetition structures (3.60)'.

6.1.4 E-PROBSOL and M-PROBSOL applications

The final part of the survey explored students' perceptions of the web-based and mobile versions of PROBSOL. Figure 4 shows the comparison of respondents' perceptions related to using E-PROBSOL and M-PROBSOL to support their learning.

Respondents perceive E-PROBSOL application more helpful than M-PROBSOL 'in learning the problem solving skills' (0.08%), and 'as an important supplement to the

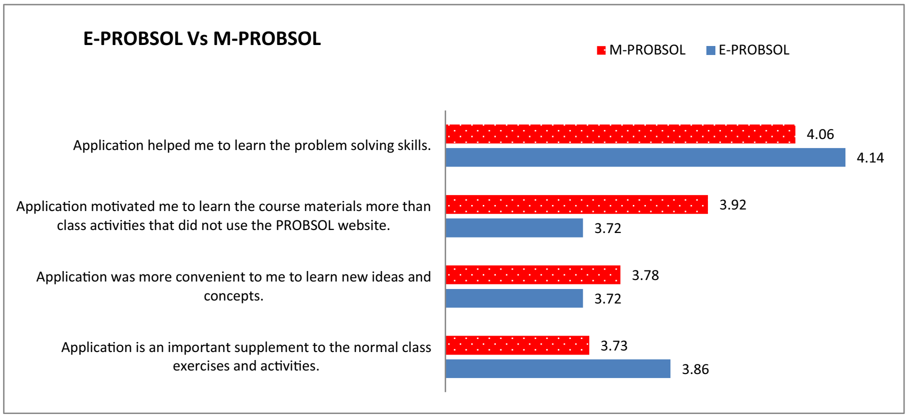


Fig. 4 Comparison of E-PROBSOL and M-PROBSOL

normal class exercises and activities' (0.13%). On the other hand, they perceive M-PROBSOL more useful than E-PROBSOL in 'learning new ideas and concepts' (0.06%), and 'motivated them to learn the course material more than class activities that did not use the PROBSOL website' (0.20%).

The second part of this section asked more questions related to E-PROBSOL and M-PROBSOL. The means and frequencies of the responses are shown in Table 3.

Respondents perceive M-PROBSOL to be more interesting (3.81), easier (3.56), quicker (3.64), friendlier (3.76), flexible (3.61), powerful (3.55), better (3.60), can be used anywhere (3.86) and anytime (3.78) compared to the E-PROBSOL application. Moreover, the respondents prefer (3.59) and suggest (3.64) M-PROBSOL more compared to the E-PROBSOL application.

6.2 Semi-structured interviews of introductory programming instructors

The second research question, the perceptions of instructors regarding the PROBSOL application in the introductory programming course, is explored in this section.

Table 3 E-PROBSOL and M-PROBSOL applications

PROBSOL web site and Mobile application	Mean	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Non response	Total
M- PROBSOL is more interesting than E-PROBSOL.	3.81	24.6%	44.6%	20%	4.6%	4.6%	1.5%	100%
		16	29	13	3	3	1	65
M-PROBSOL is easier than E-PROBSOL.	3.56	12.3%	40%	36.9%	9.2%	0%	1.5%	100%
		8	26	24	6	0	1	65
M- PROBSOL is quicker than E-PROBSOL.	3.64	7.7%	50.8%	30.8%	3.1%	1.5%	6.2%	100%
		5	33	20	2	1	4	65
M- PROBSOL is friendlier than E-PROBSOL	3.76	15.4%	50.8%	24.6%	4.6%	1.5%	3.1%	100%
		10	33	16	3	1	2	65
M- PROBSOL can be used anywhere compared with E-PROBSOL	3.86	23.1%	47.7%	20%	6.2%	1.5%	1.5%	100%
		15	31	13	4	1	1	65
M- PROBSOL can be used anytime compared with E-PROBSOL	3.78	23.1%	36.9%	32.3%	6.2%	0%	1.5%	100%
		15	24	21	4	0	1	65
I prefer M-PROBSOL to E-PROBSOL	3.59	13.8%	36.9%	40%	4.6%	1.5%	3.1%	100%
		9	24	26	3	1	2	65
M- PROBSOL is more flexible than E-PROBSOL	3.61	10.8%	53.8%	23.1%	6.2%	4.6%	1.5%	100%
		7	35	15	4	3	1	65
M- PROBSOL is more Powerful than E-PROBSOL	3.55	15.4%	38.5%	35.4%	3.1%	6.2%	1.5%	100%
		10	25	23	2	4	1	65
M- PROBSOL is better than E-PROBSOL.	3.60	12.3%	47.7%	24.6%	10.8%	1.5%	3.1%	100%
		8	31	16	7	1	2	65
I suggest M- PROBSOL to others to follow than E-PROBSOL	3.64	12.3%	47.7%	33.8%	4.6%	0%	1.5%	100%
		8	31	22	3	0	1	65

Semi-structured interviews were conducted with the IP course instructors to seek in-depth feedback on the affordances and barriers of the PROBSOL application for students. The interviews were conducted with two IP instructors. The main objectives of the interviews were to explore instructors' perceptions of:

1. Students' experience with E-PROBSOL and M-PROBSOL applications in the IP course
2. The impact of E-PROBSOL and M-PROBSOL applications on students learning in the IP course

and to explore:

3. Instructors' views on the strengths and weaknesses of the E-PROBSOL and M-PROBSOL applications

And, finally, to suggest:

4. any further enhancements to E-PROBSOL and M-PROBSOL applications in the context of the introductory programming courses.

6.2.1 First impression about PROBSOL applications

Both interviewees agreed that learning to code is a challenging task for novice programmers. Moreover, it is always a difficult task for an instructor to teach an introductory programming course to students who are unaware of the programming domain.

The interviewees indicated that the interface of the PROBSOL application does not show any launcher image which illustrates to novice programmers that this application covers problem solving skills. The instructors discussed the importance of problem solving skills and this application to the students.

The interviewees found the PROBSOL application to be an interactive way to familiarize novice programmers with problem solving skills.

6.2.2 Students experience with E-PROBSOL and M-PROBSOL applications

The interviewees found that students were satisfied with the applications because all the teaching topics were covered. Moreover, students were excited to use the applications because the applications were accessible on two different platforms including a website and a mobile application. The instructors suggested that students preferred M-PROBSOL rather than E-PROBSOL because M-PROBSOL is more flexible, easy to access and use compared to the E-PROBSOL.

The applications helped students to focus more on problem solving strategies which is one of the important skills for novice programmers to understand in the programming domain.

M-PROBSOL application was available only on Google store and the interviewees indicated that students requested the application also be available on iPhone store as well.

6.2.3 Impact of E-PROBSOL and M-PROBSOL applications on students learning

The interviewees found that the applications enhance students' interaction and collaboration in class activities. Students discussed exercises with each other and shared their ideas about the programming domain. The applications provided the correct solution for each question which helped students to critically analyze their solution. Moreover, the applications were focus-oriented because the students' mistakes (wrong steps) were highlighted in their solution which helped them to learn from their mistakes.

Students' attitude and behavior towards doing the exercises were also improved by using the applications. M-PROBSOL motivated students to finish exercises at their own convenience and time. E-PROBSOL promoted teamwork and collaboration in the class setting.

Overall, the interviewees agreed that the applications impacted positively on the students' learning in the IP course. They felt that the applications improved students' problem solving skills, programming understanding, and logic capabilities.

6.2.4 Strengths of E-PROBSOL and M-PROBSOL applications

The interviewees suggested that the applications were good supporting teaching tools in the IP course. Moreover, the applications were interactive platforms for students to learn problem solving skills. Questions related to each topic in the IP course were embedded in the applications which helped students to focus on different programming constructs.

One of the challenges for novice programmers in the programming domain is how to start solving the given problem statement. The applications provided random steps to solve the problem statement which helps the novice programmers to start thinking of a solution for the given problem statement. The steps to solve the programming problems were given randomly for each question which helped students to focus only on problem solving skills without having to be too concerned much about the programming syntax.

The applications promoted practice among students which is considered one of the important skills for novice programmers (Malik 2016b). Moreover, the applications also provide an embedded question bank to the instructors.

The applications are friendly and ease to use which helps the user to focus on the solution of the given problem statement. Having the applications available as a website and mobile app facilitates the users to practice at their own convenience.

6.2.5 Weaknesses of E-PROBSOL and M-PROBSOL applications

The interviewees found that the applications are based on pseudo-code technique. Novice programmers were not familiar with the technical terms used in the solution. The applications should provide glossary of terms or use tool tips for the technical terms. Further the mobile app is only available on Google store (for android users). It should also be available for iPhone users.

6.2.6 Comparison of PROBSOL applications with traditional teaching approaches

The interviewees indicated that most of the students are grown up with technology and they liked and engaged with technology enhanced approaches to learning rather than traditional approaches. Therefore, the PROBSOL applications are a more enjoyable mode of study for students compared to the traditional teaching approaches.

The applications provide feedback immediately by showing the wrong steps in the solution. In traditional approaches it is more challenging to provide good feedback in a timely manner.

The interviewees suggested that a mix of approaches to teaching provides improved learning opportunities for students. They found that using a traditional classroom approach to teach students at least some basic knowledge of programming and problem solving strategies first provides a good foundation for students to use the PROBSOL approach to practice their newly acquired knowledge of the programming domain.

6.2.7 Suggestions for further improvement in PROBSOL applications

The interviewees suggested a number of improvements for PROBSOL. First there should be a login feature for students. Each student will have his own account. Students can save their work and monitor their progress in solving exercises.

The PROBSOL applications should also be offered in the next level of programming course so that students are able practice more problem solving skills and acquire deeper learning of the programming domain.

Additional features such as mnemonics, accelerators and tool tips should be added to the application. These features would add shortcut keys for menu items to speed up the access process of programming exercises. Tool tips can be used to enhance the understanding of the technical terms used in the solutions of the problem statement.

Finally, the interviewees suggested that if the applications are offered as a game, it will be a more challenging and rewarding process for novice programmers. Students can compete with each other and it would be more fun for them encouraging them to practice even more.

6.3 Comparison of students' final grades

This section describes the result of Research Question 3. The third research question is as follows:

> *RQ3: What is the impact of the PROBSOL applications on the students learning in the introductory programming course?*

This question is probed by comparing the final grades of the IP course for the last two semesters (Semester 2, 2017–18 and Semester 1, 2018–19). In semester 1, 2018–19, the course was offered with the PROBSOL applications and the other semester (Semester 2, 2017–18), the course was offered with the traditional approach. The comparison of these semesters will indicate the impact of PROBSOL applications on the students' performance.

Students' were grouped into three achiever categories (high, medium and low) based on their final marks obtained in the IP course. These categories are based on the work of Malik and Coldwell-Neilson (2017b). Students who obtain final marks in between 85 to 100 are considered as high achievers; those in between 65 to 84 are medium achievers, and those students in between 50 to 64 are considered as low achievers. If students' marks are less than 50 they are considered as fail (Fig. 5).

The PROBSOL applications provide a positive improvement (3%) in the high achiever category, with 13% of the students who passed the course with the PROBSOL applications in this category compared with 10% for the traditional approach. The medium achiever category shows an increase of 3%; 50% of the students reached the medium achiever category using the traditional approach, compared with 53% for the PROBSOL applications. There is also a small positive improvement in the low achiever category of 1%; 25% of students with the PROBSOL applications compared with 24% for the traditional approach. The failure rate with the PROBSOL applications semester is 6% compared with the 9% for the traditional approach semester, a reduction of 3%. Likewise, the dropout rate (those students who do not complete the course) with the PROBSOL applications semester is 3% compared with the 7% for the traditional approach semester, and it is reduced by 4%.

7 Discussion

The PROBSOL applications were offered to enhance problem solving skills of novice programmers in the IP course. The survey was conducted with the IP students to determine the impact of applications on their learning process. Overall, the overwhelming majority of students appreciated both applications in the teaching and learning process of the IP course.

The applications were offered in two different modes of learning as a website and a mobile app. The current generation of students use digital technologies extensively in their daily lives and they prefer technology enhanced learning (Sarkar et al. 2017). The

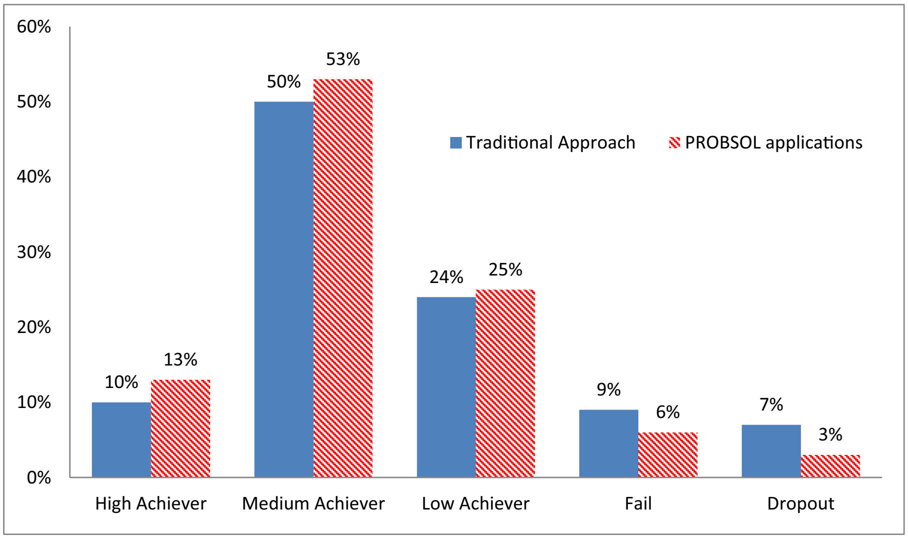


Fig. 5 Shows the comparison of students' grades in the IP course. The PROBSOL applications provide a positive improvement

current project offered learning to the students in their preferred way. Students often are bored with the traditional approach to learning programming. The application developed students' interest, affective engagement and confidence in the subject area. This finding is consistent with that suggested by Iqbal et al. (2013).

Some textbooks related to introductory programming cover problem solving strategies only in the early chapters. PROBSOL on the other hand, provided problem solving questions related to all the teaching topics of the IP course. This process provides deep learning of programming concepts to novices. Moreover, it helps novices to practice and understand programming structures in a better way. The applications provides all the steps to solve a given problem statement which helps novices to focus only on problem solving strategies.

The applications provide correct solutions of the given problem statements which helps students to compare their solution and get immediate feedback. Errors in the students' solution are highlighted in red which helps students to critically analyze their own solution.

The applications cover all the teaching topics of the IP course. Students learn basic programming concepts without worrying to much about the syntax of the programming language. This process discourages surface learning of the programming concepts and promotes deep learning, cognitive gain and engagement of different programming constructs. This finding is consistent with that suggested by Malik and Coldwell-Neilson (2017b).

The application was offered as E-learning (E-PROBSOL) and M-learning (M-PROBSOL). Students and instructors appreciated both applications in the teaching and learning process of the course. Two different modes of learning give more flexibility, opportunities and access to students in learning problem solving skills. Students can practice questions anywhere and anytime. This finding is consistent with that suggested by Iqbal et al. (2013).

Instructors considered the applications as additional supporting teaching tools in the course. Moreover, the applications are good alternative of traditional pen and paper learning approach in the course. The applications provide a collection of questions related to all topics in the IP course. Further, it promotes more practice for problem solving skills in the course. This finding is consistent with that suggested by Malik (2016b). Additional features such as glossary of terms or tool tips for the technical terms were suggested by instructors to enhance the usability of applications in the learning process. This finding is consistent with that suggested by Malik (2016a).

Students prefer M-PROBSOL more compared to the E-PROBSOL application in the learning process. Mobile learning is more flexible, available anywhere and anytime compared to the E-learning. Keegan (2012) concluded that 'mobile learning is a harbinger of future learning'. Iqbal et al. (2013) discussed that 'M-learning has some unique characteristics such as its portability, its ubiquitous nature and its individuality that make it better than E-learning in the education field'. On the other side, there are some technological shortcomings in M-learning such as small screen size, short battery life, and limited storage capacities of mobile devices (Shanthi and Al-Mukheini 2010; Hulme 2007).

The comparison of students' grades show that the PROBSOL applications impact positively on the students' learning in the IP course. The applications provide better understanding of the programming domain to the novice programmers. Moreover, it helps students to develop their interest and confidence in the programming domain and thus reduced attrition rates (failure and dropout) in the course. The applications help students to devise the solution of the problem statement without much worry about the syntax and semantics of the programming language. Overall, the PROBSOL applications impact positively on the students' achievements and performance in the course. Tawafak et al. (2018) discussed that 'the technology integrated tools emphasis on student interest in order to develop their academic performance.' (p. 2).

8 Conclusions

The PROBSOL applications were offered in two different modes of learning (E-PROBSOL and M-PROBSOL) in the introductory programming (IP) course to enhance problem solving skills of novice programmers. The survey was conducted to collect students' feedback. Semi-structured interviews were performed to collect instructors' opinion about the applications.

Students appreciated both E-PROBSOL and M-PROBSOL applications in the teaching and learning process of the IP course. The applications improve programming understanding, logic capabilities and problem solving skills of novice programmers. The applications also improve students' attitude and behavior towards doing the exercise questions. E-PROBSOL promotes teamwork and collaboration in the class setting while M-PROBSOL motivates students to finish exercises at their own convenience and time.

The applications cover all the teaching topics of the IP course and promote practice for novices. Students have to practice the problem solving skills for the whole course instead of just doing it in the early chapters. This process enhances students understanding of programming structures and promotes deep learning of programming domain.

Instructors informed that students enjoy the applications more than the traditional teaching approaches because they belong to digital natives' generation and prefer technology enhanced learning. The applications promote problem solving skills which are considered one of the important skills for novice programmers.

The comparison of students' grades show that the students' perform better in all the three achiever categories (high, medium and low) after introducing the PROBSOL applications in the course. Moreover, attrition rates (failure and dropout) were also reduced in the semester offered with the PROBSOL applications compared to the previous semester offered with the traditional approach. Overall, the PROBSOL applications impact positively on the students' achievements and performance in the course.

References

ACM-IEEE Joint Task Force on Computing Curricula,: Computer Science Curricula 2013 (2013). ACM Press and IEEE Computer Society press. https://doi.org/10.1145/2534860

Ala-Mutka, K. (2004). *Problems in learning and teaching programming – a literature study for developing visualizations in the codewitz-minerva project*, Codewitz needs analysis, retrieved 20/10/2018, http://www.cs.tut.fi/~edge/literature_study.pdf.

Australian Computer Society (2013). The ICT profession body of knowledge, retrieved 25/9/2015, acs.org.au/__data/assets/pdf_file/0013/24502/The-ICT-Profession-Body-of-Knowledge-23-Sept-2013.pdf.

De Raadt, M. (2008). *Teaching programming strategies explicitly to novice programmers'*, PhD thesis, University of Southern Queensland, Australia, retrieved 3/5/2018, USQ ePrints, https://eprints.usq.edu.au/4827/.

De Raadt, M., Toleman, M., & Watson, R. (2005). *Textbooks under inspection*, University of Southern Queensland, Australia, retrieved 5/6/2018, http://eprints.usq.edu.au/167/1/TechReport_Draft_10.pdf.

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S. J. (2015). A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. *Journal of Computer Assisted Learning, 31*(4), 345–361. https://doi.org/10.1111/jcal.12099.

Hulme, A. K. (2007). Mobile usability in educational context: What have we learnt? *International Review of Research in Open and Distance Learning, 8*(2), 1–17.

Iqbal, S., & Harsh, O. K. (2013). A self review and external review model for teaching and assessing novice programmers. *International Journal of Information and Education Technology, 3*(2), 120–123.

Iqbal, S., Chowdhury, M., & Harsh, O. K. (2013). Mobile devices supported learning for novice programmers. In *Proceeding of the 2nd international conference on E-learning and E-technologies in education* (pp. 277–282). Poland: IEEE.

Kanaki, K., & Kalogiannakis, M. (2018). Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses. *Education and Information Technologies, 23*(6), 2673–2698.

Keegan, D. (2012). *The future of learning: From eLearning to mLearning*, ZIFF papiere 119. Retrieved from https://files.eric.ed.gov/fulltext/ED472435.pdf. Accessed 20 Dec 2018.

Kölling, M., & Rosenberg, J. (1996). BlueJ - a language for teaching object-oriented programming. In *Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education*, pp. 190–194, St. Louis, USA.

Koulouri, T., Lauria, S., & Macredie, R. D. (2015). Teaching introductory programming: A quantitative evaluation of different approaches. *Transactions of Computer Education, 14*(4) ACM.

Malik, I.S. (2016a). *Role of ADRI model in teaching and assessing novice programmers*, PhD Thesis, Deakin University, Retrieved April 2017, http://dro.deakin.edu.au/view/DU:30088862.

Malik, I.S. (2016b). Enhancing practice and achievement in introductory programming using an ADRI editor. In *Proceedings of the IEEE International Conference on Teaching, Assessment and Learning for Engineering*, pp. 32–39, IEEE, Thailand, 7–9 December. https://doi.org/10.1109/TALE.2016.7851766.

Malik, I. S. (2019). Assessing the teaching and learning process of an introductory programming course with Bloom's taxonomy and Assurance of Learning (AOL). *International Journal of Information and Communication Technology Education (IJICTE), 15*(2), 130–145. https://doi.org/10.4018/IJICTE.2019040108.

Malik, I. S., & Coldwell-Neilson, J. (2016). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*. Springer. https://doi.org/10.1007/s10639-016-9474-0.

Malik, I. S., & Coldwell-Neilson, J. (2017a). Comparison of traditional and ADRI based teaching approaches in an introductory programming course. *Journal of Information Technology Education: Research, 16*, 267–283. Retrieved from http://www.informingscience.org/Publications/3793. Accessed 04 May 2018

Malik, S.I., & Coldwell-Neilson, Jo. (2017b). Impact of a new teaching and learning approach in an introductory programming course. Journal of Educational Computing Research, SAGE, vol. 55, No. 6, pp. 789–819, 2017. DOI: https://doi.org/10.1177/0735633116685852

Malik, I.S., & Coldwell-Neilson, Jo. (2018). Gender difference in an introductory programming course: New teaching approach, students' learning outcomes, and perceptions. Education and Information Technologies, vol. 23, No. 6, pp. 2453–2475. DOI: https://doi.org/10.1007/s10639-018-9725-3, Springer.

Malik, S.I., Mathew, R., & Hammood, M.M. (2019). *PROBSOL: A web-based application to develop problem solving skills in introductory programming*. In: Al-Masri A., Curran K. (eds) Smart Technologies and Innovation for a Sustainable Future. Advances in Science, Technology & Innovation, pp. 295–302, Springer.

Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. (2015). Learning to generate Pseudo-code from source code using statistical machine translation. In *Proceedings of 30th IEEE/ACM international conference on Automated Software Engineering (ASE)*. https://doi.org/10.1109/ASE.2015.36, USA.

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016). Using scratch and app inventor for teaching introductory programming in secondary education. A case study. *International Journal of Technology Enhanced Learning, 8*(3/4), 217–233.

Reardon, S., & Tangney, B. (2014). Smartphones, studio-based learning, and scaffolding: Helping novice learn to program. *Transaction on Computing Education, 14*(4) ACM.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172.

Sarkar, N., Ford, W., & Manzo, C. (2017). Engaging digital natives through social learning. *Systemics, Cybernetics and Informatics, 15*(2), 1–4.

Shanthi, D., & Al-Mukheini, T.S. (2010). Impact of mobile learning in the colleges of applied sciences in Sultanate of Oman. In *2nd International conferences on higher education and quality assurance*, 12–13 June, Muscat, Oman.

Shuhidan, S.M. (2012). *Probing the minds of novice programmers through guided learning*, PhD thesis, RMIT University, Australia.

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850–858.

Taheri, S. M., Sasaki, M., & Ngetha, H. N. (2015). Evaluating the effectiveness of problem solving techniques and tools in programming. In *Proceedings of Science and Information Conference (SAI) IEEE.* https://doi.org/10.1109/SAI.2015.7237253, UK.

Tawafak, R. M., Romli, A. B., Arshah, R. B. A., & Almaroof, R. A. S. (2018). Assessing the impact of technology learning and assessment method on academic performance: Review paper. *Eurasia Journal of Mathematics, Science and Technology Education, 14*(6), 2241–2254. https://doi.org/10.29333/ejmste/87117.

Webster, M. (1994). *Overview of programming and problem solving*, Merriam-Webster's collegiate dictionary, Tenth Edition, retrieved on 15/7/ 2018, http://computerscience.jbpub.com/vbnet/pdfs/mcmillan01.pdf.

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin, 28*(3), 17–22.

Yoo, J., Yoo, S., Seo, S., Dong, Z., & Pettey, C. (2012). Can we teach algorithm development skills? In *Proceedings of the 50th annual southeast regional conference* (pp. 101–105). New York: ACM.

Affiliations

Sohail Iqbal Malik[1] **· Roy Mathew**[1] **· Rim Al-Nuaimi**[1] **· Abir Al-Sideiri**[1,2] **·**
Jo Coldwell-Neilson[3]

[1] Buraimi University College, Al-Buraimi, Oman

[2] Universiti Tenaga National (UniTen), Kajang, Malaysia

[3] Deakin University, Geelong, Australia