Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University

MAHMOUD M. MHASHI AND Ali M. ALAKEEL
Department of Computer Science
Faculty of Computers and Information Technology
University of Tabuk, Tabuk 71491, P.O.Box 741
Kingdom of Saudi Arabia

Abstract: Learning programming skills is not an easy task as supported by many research studies. Like other students around the world, the students of the University of Tabuk struggle through computer programming courses. This research investigates and analyzes the problems faced by computer programming students at the University of Tabuk with two main objectives. The first objective is to find out whether the students at the University of Tabuk face problems in computer programming similar to the ones faced by the students in different corners of the world as reported in the literature. The second objective is to study the impact of socio-cultural and environmental factors on learning computer programming skills by the students of the University of Tabuk. To perform our research, a case study and a questionnaire were designed. In order to ascertain whether our students face difficulties in programming, a case study with 10-questions test was performed. The results have shown that our students are in real difficulties. In the questionnaire, the questions reflect a wide range of views pertaining to educational facilities such as curriculum and teaching materials, lab equipments and class rooms’ environment, and the adequacy and proficiency of the professors and teaching assistants. This paper analyses the results of the questionnaires. The questionnaires provide insight into the environmental and socio-cultural effects and the difficulties experienced while learning and teaching programming.

Key-Words: Programming difficulties, learning, teaching, novices, and environmental and socio-cultural effects.

1 Introduction

Computer programming skills are very fundamental to the computer related fields of studies. Most of the students are required to take an introductory programming course and the basics of it. Programming is an academic discipline. Furthermore, programming is a skill requiring novice programmers to utilize multiple types of learning simultaneously [1].

Novice programmers lack the knowledge and skills of programming experts. The knowledge of novices tends to be context specific rather than general [2]. Winslow [3] have concluded that novices are limited to surface and superficially organized knowledge, lack detailed mental models, fail to apply relevant knowledge, and approach programming “line by line” rather than using meaningful program “chunks” or structures. Novices spend little time in planning and testing code, and try to correct their programs with small local fixes [4].

The computer programming learning is very complicated for many novice students at university level [5]. Computer programming is not difficult only because of the abstract concepts, but also students have problems in different issues related to program construction. The main source of difficulty does not seem to be the syntax or understanding of concepts, but rather basic program planning [6].

The introductory programming courses have a relatively high fail rate. Hagan in [7] indicates that programming was considered the most difficult and least interesting subject by most first year students in all computing courses. Many institutes report dropout rates of 20-40 percent [8-14]. Because of the importance of computer programming and the difficulties faced by the novice students, many authors have investigated this topic in one form or another [15-27].

The organization of this paper is as follows. Section 2 contains the design, the collected data, and the analysis of the 10 questions to test the current level of programming for our students. The survey design and methodology will be introduced in Section 3 and the analysis of the results in Section 4. The results of both the case study and the survey
will be discussed further in Section 5, followed by the conclusions in section 6.

2 Testing students’ abilities case study
In order to make sure that our students facing difficulties in computer programming, they were tested on their ability to select the correct answer from a set of possibilities.

2.1 Questions Design and Results
The questions design covers three styles of questions which include 1) the prediction of the outcome of a short piece of code and choice of the correct option, 2) the selection of the correct functionality of a given piece of code from a set of possible options, and 3) the selection of one or more missing code segments from a set of possible segments of code suitable for the required functionality according to the description provided to the students.

The data was collected by asking students to answer 10 multiple choice questions. The programming code was given in C#. The questions were intended to test students’ knowledge and skills for generic, loops, and iterative processes on arrays. The students volunteered to participate in this case study. The students were allowed to make drawings and calculations on their respective questions papers. The students answered all the questions in 25 to 65 minutes. The students were not trained earlier therefore they were not well prepared to do well on the type of multiple choice questions used in this study.

The following is a statistical analysis of the performance data. Figure 1 shows a histogram of scores (out of 10) of 165 students who were given all ten questions. The scores cover the complete range, from 0 to 10. Table 1 shows the boundaries for these students.

![Figure 1: Distribution of scores for students who were given all 10 Questions (N=165).](image)

<table>
<thead>
<tr>
<th>Range No.</th>
<th>Score Range</th>
<th>No. of students</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 - 3</td>
<td>123</td>
<td>75%</td>
</tr>
<tr>
<td>2</td>
<td>4 - 6</td>
<td>36</td>
<td>22%</td>
</tr>
<tr>
<td>3</td>
<td>7 - 10</td>
<td>6</td>
<td>3%</td>
</tr>
</tbody>
</table>

Table 1: The boundaries for students who were given all 10 Questions (N=165).

2.2 Questions Discussion
Table 2: shows the percentage of students who answered each of the 10 questions correctly. The relative difficulty of each question can be defined by the percentage of students who answered the question correctly. The hardest questions are 1, 2, 5, and 7. At first glance, question 1 might appear to count the number of elements added in the summation of the elements in the array, which is value 5 (choice d). However, on closer inspection of the code, it can be seen that the elements in the array will be added till the total becomes equal to the value of size, so the correct answer is 3 (choice c).
Table 2: Percentage of students who answered each of the 10 Questions correctly.

<table>
<thead>
<tr>
<th>Question no.</th>
<th>Percentage Correct</th>
<th>No. of Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>09%</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>16%</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>18%</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>24%</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>06%</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>24%</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>09%</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>33%</td>
<td>55</td>
</tr>
<tr>
<td>9</td>
<td>21%</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>30%</td>
<td>50</td>
</tr>
</tbody>
</table>

The same idea is repeated in question 7 in another form. In Question 2, the code might appear to count the number of swaps between the smaller elements in the first array with the corresponding elements in the second array, which is 2 (choice C). However, it can be seen that the elements at position 0 in the arrays are not reached, so the correct answer is 1 (choice D). In question 5, it might appear to test whether the data in the array are sorted in descending order. Most students got confused between the answers (choice a, and choice e). On closer inspection, the function tests whether the data in the array is sorted in descending order or not, so the correct answer is choice c. So the ability to read what the code actually does, rather than what we think it should do, is an important programming skill.

Most of the students (75%) who fall within the first range, scoring 0-3 out of 10, we think they might suffer from problems in fundamentals, an inability to problem-solving, and language constructs. While these statistics will surprise many lecturers, yet these are very useful statistics for them to quote to their students. The students’ weakness was the inability to reason at a more abstract level to select the missing line of code. It seems that the students in this range answer the questions based on their look rather than their meaning. Furthermore, they are very weak in English. This is because in question number 3, the word “including” is used in a wrong answer choice (e) and the word “excluding” is used in the correct answer choice (d). However, most of the students selected the wrong answer choice (e). This might have happened because the students do not understand the meanings of these two words or they got confused in their meanings. Based on the previous, it is not possible to draw any firm conclusions as to the exact cause of this weakness. To explore that issue further, the rest of the paper examines the other survey gathered in this project.

3 Survey and Design Methodology

In this survey, we studied the C# course to get ideas for topics and information of the preferred materials, and to find out the perceptions on the different programming phases and learning situations. The questionnaire had six different sections: course contents, programming environments, student environments, teaching aspects, materials, and instructor issues from student point of view. The questions can be seen in Appendix II.

The goal of the first section was to find out difficulties in learning the course contents, including the issues in program construction and the programming concepts. The questions asked the respondents to grade programming aspects and concepts on a five-point scale from least difficulty (1) to the most difficulty (5). The second section contained questions about programming environment issues. The goal was to find out what kind of programming environment issues the students consider most effective. The scale for responses was the same as before, from 1 to 5, varying from the student feeling that he was learning never in that kind of issue (i.e., 1 is the minimum) to learning always (i.e., 5 is the maximum).

The third section contained questions about student environment issues. The goal was to find out the effect of students’ environment on learning programming. The scale for responses was from 1 to 5, varying from the least positive effect to the most of positive effect on learning programming. Section number four contains questions about teaching programming issues. Responses are varying from 1 (minimum positive effect) to 5 (maximum positive effect) on learning programming. The fifth section contained questions about the materials. The goal was to find out the effect of teaching material on the learning programming. The scale for responses was varying from 1 - practically useless to 5 - very useful. The last section contained issues about the lecturer from the students’ point of view. The goal was to find out the effect of the lecturer behavior and other issues on the student achievement. The scale of responses was varying from the least positive effect to the most positive effect.
4 Survey Result Analysis

In all 143 students answered the survey. The results of the different sections, including course contents, programming environments, student environments, teaching aspects, materials, and instructor issues from student point of view about learning programming are presented in Table 3. The respondents perceived as the most difficult programming concepts were loop structures (C5), recursion (C6), arrays (C7), pointers (C8), passing parameters (C9), abstract data types (C10), and using the library functions (C14). Recursion, pointers, and abstract data types are abstract concepts and thus many novice programmers find them difficult to understand.

In the programming environment, the students rated access to computers labs (PE2), understanding the programming syntax (PE4), designing programs (PE5), Dividing problems into sub-problems (PE6), designing functions (PE7), and finding bugs (PE9) the most difficult. Some students wrote that there are no free computer labs available at the campus for practice.

The least positive effect of student environment issues were the student's opportunity to ask questions (SE2), programming experience before attending this course (SE4), and attending programming exercise (SE5). Regarding asking questions, as far as we know, the opportunity to ask questions is available. However, most students might hesitate to ask questions during the lecture. About the third issue, some students wrote that there was no time reserved to solve exercises other than the time of lectures.

Table 3: Results are on the different sections about programming learning.

<table>
<thead>
<tr>
<th>Code</th>
<th>Level of difficulties of programming concepts on programming learning</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Declaration</td>
<td>1.8</td>
</tr>
<tr>
<td>C2</td>
<td>Variables</td>
<td>1.9</td>
</tr>
<tr>
<td>C3</td>
<td>Assignment statement</td>
<td>2.1</td>
</tr>
<tr>
<td>C4</td>
<td>Conditional statements</td>
<td>2.3</td>
</tr>
<tr>
<td>C5</td>
<td>Loop structures</td>
<td>3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Programming Environment effect on programming learning</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C6</td>
<td>Recursion</td>
<td>3.3</td>
</tr>
<tr>
<td>C7</td>
<td>Arrays</td>
<td>3.2</td>
</tr>
<tr>
<td>C8</td>
<td>Pointers</td>
<td>4.1</td>
</tr>
<tr>
<td>C9</td>
<td>Passing Parameters</td>
<td>3.2</td>
</tr>
<tr>
<td>C10</td>
<td>Abstract data types</td>
<td>3.0</td>
</tr>
<tr>
<td>C11</td>
<td>Input/output handling</td>
<td>1.9</td>
</tr>
<tr>
<td>C12</td>
<td>Syntax errors</td>
<td>2.0</td>
</tr>
<tr>
<td>C13</td>
<td>Logical errors</td>
<td>2.3</td>
</tr>
<tr>
<td>C14</td>
<td>Using and understanding library functions</td>
<td>3.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Student Environment effect on programming learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE1</td>
<td>Level of effort you put into the course</td>
</tr>
<tr>
<td>SE2</td>
<td>The student's opportunity to ask questions</td>
</tr>
<tr>
<td>SE3</td>
<td>Lecture attendance</td>
</tr>
<tr>
<td>SE4</td>
<td>Level of programming experience before this course</td>
</tr>
</tbody>
</table>
### Attending programming exercise

| SE5 | Attending programming exercise | 2.0 |

### Solving programming exercise, homework’s and assignments

| SE6 | Solving programming exercise, homework’s and assignments | 3.6 |

### Teaching Issues effect on programming learning

| TI1  | Lectures | 3.6 |
| TI2  | Exercise sessions | 1.8 |
| TI3  | Practical sessions | 3.0 |
| TI4  | Studying alone | 2.6 |
| TI5  | Studying with friends | 3.3 |
| TI6  | Studying with help from someone knows programming | 3.9 |

### Materials Effect on programming learning

| ME1  | Course book | 2.7 |
| ME2  | Lecture notes | 3.0 |
| ME3  | Web-pages | 2.6 |
| ME4  | Copies of transparencies | 2.4 |
| ME5  | Exercise questions and answers | 2.8 |
| ME6  | Programming examples | 3.0 |
| ME7  | Interactive visualizations | 3.0 |
| ME8  | The appropriateness of the amount of material the course attempted to cover | 3.1 |
| ME9  | The content is arranged in a clear, logical and orderly manner | 3.4 |
| ME10 | The content covers most of the topics you expected to find | 3.4 |
| ME11 | The level of usefulness of examples was shown | 2.4 |

| Instructor from student point | |

### Of view

| 11  | Instructor knowledge of programming | 3.0 |
| 12  | The instructor returned examinations and papers in a timely fashion | 3.5 |
| 13  | The instructor provided feedback regarding completed course assignments | 2.5 |
| 14  | The level of benefit of the feedback you received | 2.6 |
| 15  | The size of the class | 3.1 |
| 16  | The instructor's responsiveness to student questions and ideas | 2.4 |
| 17  | The instructor's presentations impacted on my understanding of material | 3.2 |
| 18  | The instructor's ability to communicate the subject to the student | 3.1 |
| 19  | Availability of instructors (office hours, e-mail and telephone calls) | 3.2 |
| 110 | The instructor's enthusiasm in talking about course material | 3.1 |
| 111 | The instructor's interest in teaching | 3.2 |
| 112 | The instructor's use of examples or personal experience to help get points across | 3.1 |
| 113 | The instructor's ability to relate the course concepts in a systematic manner | 2.8 |
| 114 | The instructor is able to teach at the students' level | 3.0 |
| 115 | Was the use of the board/power-point useful | 3.5 |

The least useful teaching issues were exercise sessions (TI2) and studying alone (TI4). This supports SE5 where there was no reserved time to solve exercises. Students rated studying with friends (TI5) and studying with help from someone who knows programming (TI6) as the most useful.
the help of someone who knows programming (TI6) more useful than studying alone (TI4).

The students were happy from the benefit they got from the lectures (ME2). This was because; they rated the usefulness from the lecture notes much higher than it was from course book (ME1), Web-pages (ME3), or copies of transparencies (ME4). The students rated the exercise questions and answers (ME5) less useful than as it was supposed to be. The usefulness of examples presented in the class (ME11) was not supported enough by the students understanding programming.

Regarding the section of the instructor from the students point of view, the students were not happy with the feedback provided by the instructor regarding the completed assignments (I3). Furthermore, they were not satisfied with the usefulness of the provided feedback (I4). Similarly, the students were not convinced that the instructors were not able to relate the course concepts in a systematic manner.

5. Discussion

First of all we have to bear in mind that the responses are subjective opinions of the students who answered the questions in the survey. The students do not always see their difficulties. The students might think that they well understand a specific topic but actually they do not. In the survey, the level of difficulties for loops (C5) with average (3.0) and arrays (C7) with average (3.2) were not as the same level as pointers (C8) with average (4.1). On the other hand, most of the students (75%) who answered the 10 questions (Table 1) fall within the first range, scoring 0-3 out of 10. So, one of the bigger problems of students does not seem to be the understanding of the basic concepts but rather learning to apply them. The results show that there were many structures related aspects that are difficult for the students to learn which include loops, recursion, arrays, pointers, passing parameters, and the abstract data types (Table 3). The theoretical lectures are very important in learning programming; students need practical sessions to understand the concepts.

The students admitted that there were some difficulties in programming environment. These difficulties ranged from access to computer labs to finding bugs (see Table 3). However, the frequent practical exercises that are a feature of most programming courses are central in helping understanding the language features. Programmers must learn to develop tracking and debugging skills so as to model and correct their programs. In practical/laboratory based work it may be useful for instructors/teaching assistant to particularly attend and involve as in aid to designing the problems and assisting individual students to change their code.

Student environments had a major effect on learning programming. It is clear from the survey that the students, even though they put in a serious effort on the course (SE1), (SE3), and (SE6), but still the results in the 10-questions were very low. Furthermore, the students’ opportunity to ask questions was low (SE2). This might have happened because the students were afraid and hesitant, they had lots of questions but they did not know what they had to ask, or the lecturers lacked the skills to encourage the students to ask questions.

The experience of the students was low (SE4). The students were weak in English language. This was clear from their answers to the question number 3 as most of the students selected the wrong answer ‘E’. When we asked some students about that, we realized that the students did not understood the meaning of the word ‘Excluding’ in the correct answer ‘D’ and the meaning of the word ‘including’ in the wrong answer ‘E’. So, this supports the results in the literature in that the students whose first language was not the same as that used in the course, performed less well than other students. The English background is in relation to the students’ performance in CS course and in CS degree in general. Poor English skills are related to the difficulties and correspondingly strong English background is related to the success both in CS course and in degree level.

One of the problems in teaching programming is the exercise sessions (TI2). The students seem to be dissatisfied with their effectiveness. The exercise sessions need more attention. We think the students overestimate achievement from the lectures and the practical sessions. This is the reason why students prefer studying with friends or seeking help from paid home tutors who are good at programming instead of studying alone.

The students appear to consider the lecture notes more useful than the course book (written in English language), Web-pages, and copies of transparencies. This lends support to the fact that the students were weak in the English language. The rest of the material forms were considered equally useful. Thus the students want more attention to be given to some other kind of materials.

The students were not happy with the feedback provided by their lecturers on completed course assignments (I3). In addition, if that feedback was available it was not as useful as it was supposed to
be (I4). It is necessary that the lecturers give enough feedback about the exercises and assignments and solve them. This might help the students to understand how the programs work and understand their code better. Increasing the number of exercises, assignments, carefully selecting them, and solving them after completion will increase the students’ knowledge and their skills’ development.

The students were also not happy with the instructor's response to the students’ questions and ideas (I6). The lecturers must have experience in teaching programming and must have the ability to affect and control the teaching interaction in order to make interaction process more effective to help the students learn programming.

6. Conclusion

Learning to program is a complex task. This paper has presented results from a case study and a questionnaire whose purpose was to make sure that our students face difficulties in programming and to rank programming issues in order of difficulties from students’ points of view. The students face real difficulties in programming. The students lack the ability to problem solving and design at its different stages, and general programming topics. Most of the students lack skills even to analyze a short piece of code. In addition, students are not hard working and they lack understanding. Poorly designed course, students' weakness in English language, lack of practice and exercises feedback, and lecturers insufficient skills in organizing the material and lack of support by the university (large number of students in labs, and availability of assistant lecturers), were seen as possible factors related to the difficulties. Finally, one of the difficulties was the students’ level of maturity. This category included students’ conduct: need for hard work, class and lab attendance, solving home works and exercises, getting feedback from the lecturers, and involvement in class interactive visualizations.

7. REFERENCES


**Appendix (1)**

Faculty of Computers and Information Technology
Teaching Evaluation Questionnaire

Please circle the correct answer for each one of the following ten questions.

Q1. You are given the following segment of code:

```
int[] arr = {1, 2, 3, 4, 6};
int size = 6;
for (int count= 0, total = 0; total < size; count++)
total += arr[count];
```

What is the value of the variable “count” after executing the above code?

a) 4  b) 2  c) 3  d) 5  e) 6
Q2. You are given the following segment of code:
```java
int[] arr1 = {5, 3, 10, 4};
int[] arr2 = {6, 2, 9, 7};
int i = arr1.length-1;
int j = arr2.length-1;
int count = 0, temp;
while (i > 0) {
    if (arr1[i] < arr2[j]) {
        count++;     temp = arr1[i];       arr1[i] = arr2[j];
        arr2[j] = temp;  
    }
    i--;  j--; }
```
After the execution of the above code, the value of “count” is:

a) 5  b) 4  c) 2  d) 1  e) 9

Q3 What is the function of the following segment of code?
```java
int grade, sum = 0;
for ( int i = 1;  i<= 10; ++i) {
    grade = Console.ReadLine();
    while ( grade < 35)
    grade = Console.ReadLine();   
    sum += grade;
}
```

a) Find the sum of 10 grades  

b) Find the sum of grades from 1 to 10  

c) Find the sum of grades < 35  

d) Find the sum of 10 grades excluding grades < 35  

e) Find the sum of 10 grades including grades < 35  

Q 4.
You are given the following segment of code:
```java
int [] arr1 = {2, 4, 4, 2, 2, 4};
int [] arr2 = {0, 0, 0, 0, 0,0};
for ( int i = 0; i < arr1.length; ++i )
    arr2 [ arr1[i] ] = 1;
int count = 0;
for (int i = 0; i < arr2.length; ++i )
    if ( arr2[i] == 1 ) ++count;
```

After the execution of the given code, what is the value of “count”:

a) 1  b) 2  c) 3  d) 4  e) 5

Q 5. The following method “isDescending” returns true if the array is sorted in descending order and return false Otherwise.
```java
public static boolean isDescending (int []arr) {
    //missing code  
}
```
The missing code is:

(a) for (int i=0;i<arr.length-1;i++)
    { if (arr[i] < arr[i+1] ) return false;  }
    return true;

d) boolean flag = false;
    for (int i=0;i<arr.length-1;i++)
    { if (arr[i] < arr[i+1] ) flag = true;  }
    return flag;

e) boolean flag = true;
    for (int i=0 ; i<arr.length; i++)
        { if ( arr[i] < arr[i+1] ) flag = false;
          else flag = true;  }
    return flag;

Q6 Consider the following code fragment:
```java
int x = {6, 4, 2, 1};
int temp;       int j = 0;
int i = x.length-1;
while (i > j)
{  temp = x[i]; x[i] = x[j];     x[j] = temp;  
    i--;  j++;   }
```
After this code is executed, array “x” contains the values:

a) {2, 4, 4, 6}  
b) {6, 4, 4, 2}  
c) {12, 8, 2, 1}  
d) {1, 2, 8, 12}  
e) {2, 4, 8, 12}

Q7 You are given the following segment of code:
```java
int[] arr = {2, 2, 3, 4, 4, 4};
int limit = 4,  i = 0, size = arr.length, total = 0;
while ( (total<limit) && (i<size) )  {
    total += arr[i];   ++i;   }
```
The function of the above code is:

a) Sum all the values in the array.  
b) Sum the first 4 values in the array.  
c) Sum values in the array till it finds the value 4.  
d) sum the values in the array till the last occurrence of 4  
e) sum the values in the array till total becomes 4.

Q 8. Any two numbers in an array of integers are called inversion if the number that occurs first in the array is larger than the number that occurs next in another place.

For example, consider an array “arr” that contains the following 4 numbers:
```
4 5 3 2
```
There are 5 inversions in that array, as:
```
arr[0]=4 > arr[2]=3  
arr[0]=4 > arr[3]=2  
```
The code fragment below is intended to count the number of inversions in an array “arr”:
```java
int Count = 0;
for ( int i=0 ; i<arr.length-1 ; i++) {
    for /*missing code*/
    { if ( arr[i] > arr[i+1] )
        ++Count;  
    } 
the /*missing code*/ is:

a) ( int j=0;   j<arr.length;   j++)

```
b) ( int j=0; j<arr.length-1; j++ )
c) ( int j=i+1; j<arr.length; j++ )
d) ( int j=i+1; j<arr.length-1; j++ )

Q9 You are given the following segment of code
int count = 0;
for (int i = 5; i<= 19; i+= 7)
for (int j = 1; j < 2; j += 4)
count++;
After the execution of the above code the value of “i”, “j”, and “count”, respectively are:
a) 19, 2, 3       b)19, 1, 3       c)12, 5, 3
d) 26, 5, 3     e) 26, 5, 38

Q 10.
The segment of code below is intended to copy into an array of integers called “arr2” any numbers in another integer array “arr1” that are multiple of 7. For example, if “arr1” contained the numbers:
4, 14, 6, 21, 1, 7
then after the copying process, “array2” should contain:
14, 21, 7
int j = 0;
for ( int i=0 ; missing_cod_1 ; i++ )
{
   if ( arr1[i] % 7 == 0 )
   {
      missing_cod_2; missing_cod_3;  
   }
}
The missing code “missing_cod_1”, “missing_cod_2” and “missing_cod_3” respectively are:
a) i<arr1.length
++j; arr2[j] = arr1[i]
b) i<arr1.length
arr2[j] = arr1[i]
++j
c) a1<=array1.length
array2[a2] = array1[a1]; ++a2
d) a1<=array1.length
++a2
array2[a2] = array1[a1]