Decoding Doodles: Novice Programmers and Their Annotations

Jacqueline Whalley  
School of Computer and Information  
Sciences  
Auckland University of Technology  
Private Bag 92006, Auckland 1020,  
New Zealand  
+64 (9) 921 9999  
jacqueline.whalley@aut.ac.nz

Christine Prasad  
School of Computing and Information  
Technology  
Unitec New Zealand  
Private Bag 92025, Auckland, New Zealand  
+64 (9) 815 4321  
cprasad@unitec.ac.nz

P. K. Ajith Kumar  
Bay of Plenty Polytechnic  
Tauranga,  
New Zealand  
+ 64 (7) 5440920  
Ajith.Kumar@boppoly.ac.nz

Abstract

This paper reports on the annotations made by novice programming students on an exam script. We investigate the level of reasoning that the students achieve when answering a short answer question using the SOLO taxonomy and relate this to the type and number of annotations they made. The questions and annotations were classified and the relationship between question type, student performance and the tendency to annotate was explored.

Keywords: Computing education, novice programming, annotation, doodle, cognitive taxonomies.

1 Introduction

It is no secret that, despite the best efforts of teachers in our discipline, students continue to face many challenges when learning computer programming. Recently, multi institutional multi national (MIMN) studies, as described by Fincher et al. (2005), have provided the CSEd community with a new means of gaining insight into the issues faced by novice programmers. MIMN studies such as “the McCracken Group” (McCracken et al. 2001), “the Leeds Group” (Lister et al. 2004), Bootstrapping (Petre et al. 2003), Scaffolding (Fincher et al. 2004) and BRACE (Fincher et al. 2005) have highlighted the difficulties that novice programmers face when learning computer programming. Within the CSEd community MIMN studies have facilitated hypothesis generation and data collection. They have also strengthened research methodology through a sharing of ideas and knowledge.

In this paper, we report on an analysis of data gathered by an MIMN study, the BRACElet project. This project was designed to investigate the reading comprehension skills of novice programmers. The data analysis presented in this paper focuses on the annotations or “doodles” that accompanied student responses to multiple-choice questions (MCQ’s). In particular, we aim to answer the following questions:

- Is there a relationship between students’ reasoning skills according to the different levels of the SOLO taxonomy and their use of doodles?
- Is there a relationship between student achievement and their frequency of doodling?
- Is there a relationship between the level of difficulty of a question and the likelihood of doodles being used to solve the problem?
- Are students more likely to doodle on certain types of questions?

In the discussion that follows we put the study into context, provide some insights into the use of doodles by novice programmers and in doing so identify future avenues for investigation.

2 Background

The Leeds Group defined a “doodle” as diagrams and annotations that experienced programmers write or draw when faced with the task of determining the function of the code. With respect to student responses to MCQs, they referred to a “doodle” or annotation as any marking a student makes on his or her exam paper. They categorised the doodles into 12 classes. This group found that if students traced through the code, they were more likely to get the correct answer.

McCartney et al. (2004) further analysed the Leeds Group’s data using the same categorization of doodles as the original study. They looked at the interrelationships between the kinds of annotations used by students, the difficulty and type of individual questions, and the institutions where the students were tested. They found that performance improved when students annotated their tests. Overall, they concluded that any annotation was better than none.

Fitzgerald, Simon and Thomas (2005) identified 19 strategies that novice programmers use to solve MCQ’s.
One of the strategies was doodling. They made a clear distinction between tracing code and other doodles. They claimed that “tracing is an entire process that may or may not involve doodling”. All students in their study employed a range of strategies, and the choice of strategy adopted was influenced by the problem type. Moreover, students often employed strategies poorly, which they considered an indicator of fragile knowledge.

For the purposes of our study we developed a new set of problems and designed the test script so as to encourage students to annotate them. The problem set used contained MCQ’s and short answer questions. These questions were classified within a cognitive framework that was based on the revised Blooms taxonomy (Anderson et al. 2001) and an adaptation of the SOLO taxonomy (Biggs 1999). The responses to the two short answer questions were analysed using SOLO to determine the reasoning ability of each of the students in the study. Additionally, the student responses to the MCQs were compared with expert responses from teachers of programming. Information on the project and its question set, administration and analysis has been documented in Whalley et al. (2006), Thompson et al. (2006), and Lister et al. (2006).

3 Data Collection

The data collected consisted of student answers to nine MCQs and two short answer questions. Analyses of the MCQs (Whalley et al. 2006), the short answer question 10 (Whalley et al. 2006, Lister et al. 2006) and the classification question 11 (Thompson et al. 2006) have already been reported. In this paper we are investigating the use of annotations on the student scripts.

This analysis looks at the scripts of 71 students from a single institution that participated in this study. These students had either nearly completed or just completed the first semester of their first programming course. In all cases the problem set counted towards the student’s final grade and were taken under examination conditions in a single session.

The students were encouraged to annotate their scripts and plenty of space was provided on the problem sheet with each problem appearing on a separate sheet to allow for doodling.

4 Data Coding

The doodles on the students’ scripts were analysed by the authors using a set of defined categories. The coding schema (Table 1) used for the classification of the student script annotations was a modified version of the schema described by McCartney et al. (2004).

As a result of the inclusion of new types of questions (for a full discussion of question types see section 5.2) a new doodle type emerged that is a ‘range doodle’ (R). This type of doodle was observed only for question 4 where students were required to determine whether or not a number was within a given range (see Appendix). This doodle helps identify numbers that are valid within a given range and seem to occur with conditional statements that include a Boolean operator. An example from a student script is shown in Figure 1.

![Figure 1: A category ‘R’ doodle](image)

Some of the categories used by McCartney et al. (2004) were not of interest for this analysis for example the category E (Extraneous Marks, marks that appear ambiguous or meaningless) was not used. Extraneous marks in our coding were treated as blanks.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Synchronised trace</td>
<td>S</td>
<td>Shows values of multiple variables changing, generally in a table</td>
</tr>
<tr>
<td></td>
<td>Trace</td>
<td>T</td>
<td>Shows the values of a variable as it changes (more than 1 value for at least 1 variable)</td>
</tr>
<tr>
<td></td>
<td>Odd Trace</td>
<td>O</td>
<td>Appears to be a trace but neither S nor T</td>
</tr>
<tr>
<td>Med</td>
<td>Keeping Tally</td>
<td>K</td>
<td>Some value being counted multiple times (specific variable not indicated)</td>
</tr>
<tr>
<td></td>
<td>Number</td>
<td>N</td>
<td>Shows a single variable value, most often in comparison</td>
</tr>
<tr>
<td></td>
<td>Position</td>
<td>P</td>
<td>Picture of correspondence between array indices and values</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>R</td>
<td>A doodle that depicts a range of valid and invalid values</td>
</tr>
<tr>
<td></td>
<td>Computation</td>
<td>C</td>
<td>An arithmetic or Boolean computation.</td>
</tr>
<tr>
<td>Low</td>
<td>Alternate answer</td>
<td>A</td>
<td>Student changed their answer</td>
</tr>
<tr>
<td></td>
<td>Ruled Out</td>
<td>X</td>
<td>One or more alternative answers have been crossed out. The answer appears to have been arrived at by elimination</td>
</tr>
<tr>
<td></td>
<td>Underlined</td>
<td>U</td>
<td>Part of a question underlined for emphasis</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>B</td>
<td>Either no annotations for this question or extraneous markings such as dots, lines and so forth.</td>
</tr>
</tbody>
</table>

Table 1: Annotation Categorisation
For the purposes of our analysis a further layer of classification, namely rank, has been added. The ranking allows for the clear separation of doodles based on our evaluation of the doodles importance in relation to program comprehension. This ranking provides a distinction between tracing and other doodles.

The ranking separates highly meaningful doodles and moderately meaningful doodles from low-relevance doodles. In this context low-relevance doodles are the result of an action taken after the thought process has been completed. They record the result rather than the process (for example U, X, A). On the other hand, meaningful doodles illustrate the process of thinking and are written as the thinking takes place and assist the thought processes that construct an answer.

McCartney et al. (2004) provided a similar layer of abstraction for their coding: (Blank, Some Tracing (S, T and O), Elimination (A or X), Other (everything else)). Their layers of abstraction were constructed to create disjoint classes of doodles. Some of the categories included in their “Other” layer have a moderate degree of usefulness as a program comprehension doodle. Others do not, for example U and E and this is why we developed a new grouping of doodles.

5 Results

If every student had doodled once on every question there would have been 639 annotations. When we counted any combination of doodle types on a single question as one annotation, the total number of observed annotations was 289 giving a frequency over all the questions of 44.5%.

McCartney et al. (2004) found that the percentage of questions with annotations varied from 28% to 92% over the 12 participating institutions in their study. The percentage of doodles obtained in our study was lower than that obtained in 10 of the 12 institutions in the McCartney study. This was despite the fact that the students in our study had been encouraged to annotate their scripts and had been taught synchronised tracing as a formal process. However, we should not be surprised by student reluctance to adopt tracing and code annotation as a technique in light of Thomas, Ratcliffe and Thomasson’s experiences. In 2004 they reported on an occasion where, in an exam, students were encouraged to doodle and yet almost two thirds of the students turned in a paper with no annotations.

On most occasions where our students did doodle they tended to use a single type of doodle (350 times, 60%) and using more than two types in combination was rare (12%). Figure 2 shows the frequency with which combinations of doodle types were used.

The most commonly used doodle combination was N and T. The combination of NT was used in isolation 60% of the time otherwise it was observed in combination with other type(s)

5.1 Performance and doodles

The Leeds group concluded that performance in both FC (fixed code) and SC (skeleton code) questions improves when students annotate their tests. We compared the overall achievement of our students on the 9 MCQs with the extent to which they used doodles in order to further investigate the relationship between doodling and achievement (Figure 3).

The extent to which students in the top quartile of achievement used doodles was compared with the use of doodles by students in the second and third quartiles and students in the bottom quartile. Only ten students failed to doodle on any of the questions.

Six of the students who failed to annotate at all were in the bottom quartile and 4 were in the middle two quartiles. All students in the top quartile had made at least one annotation.

Data for students in the middle two quartiles were combined because a large number of scores fell at the median score of 6 and the two quartiles could not be meaningfully separated.
A one-way ANOVA of the three groups (F = 4.48, p < 0.02) showed that higher achievers were more likely to doodle. This provides support for the conclusions reached by McCartney et al. (2006).

Students in the lower quartile averaged 2.35 doodles, the middle quartiles 4.26 and top quartile 4.88. However, the only statistically significant difference (Tukey HSD test) was between the top quartile and the bottom quartile (p < 0.01). The difference between the bottom quartile and the middle two quartiles very nearly reached the 0.05 level of significance, but the difference between the middle quartiles and the top quartile was well short of significance. A significant but weak correlation r = 0.282 (< 0.02) was found between the number of doodles and student scores.

5.2 Replicated Leeds Questions

Two of the MCQs used in this study (problems 1 and 9; see Appendix) were taken directly from the Leeds working group instrument (5 and 2 respectively; Lister et al. 2004). These two questions were included to enable some direct comparative analysis.

As reported previously (Whalley et al. 2006) the students found question 1 to be the third easiest question in the set while question 9 was the fifth easiest question. Student performance on these two questions was comparable with that observed by the Leeds group.

Seventy-three percent of our students doodled on question 1. This was by far the highest usage of doodles when compared with the other questions. Similarly in the Leeds study they found that 88% of their students doodled on this question and that it elicited the highest number of doodles.

Twenty-eight of our students used some form of trace and 6 used the formally taught synchronised trace (S). Most students who answered this question used a combination of doodle types. Of those students who used tracing only 3 used tracing in isolation. Question 1 was the only question where tracing was the predominant form of doodle type. This use of tracing as an annotation demonstrated that students could effectively trace code but they chose not to employ it as a strategy to the same extent for any other question.

Fifty-six percent of our students doodled on question 9 but 79% of students in the Leeds study doodled on this question (Table 2). Thirty-six percent (26 students) of our students used some form of high-ranking trace doodle. We have no definitive explanation for the difference observed on this question. Perhaps our students doodled less because they met this question in the second half of the problem sheet.

5.3 Question types and doodle style

The BRACElet problem set contained four types of questions, two of which had been previously defined as fixed code and skeleton code (McCartney et al. 2004). We introduced two new question types that we defined as change of representation and change of logic.

Fixed code (FC) problems are defined as questions where a student is given a code fragment and asked to predict the result after executing that code.

Skeleton code (SC) problems are defined as questions where the student is given a code fragment and asked to complete the code so that it will perform a specified task.

Change of representation (CR) problems are defined as questions where the student is given a code fragment and is asked to identify the same program in an alternative representation or visa versa. For example, question 2 in our problem set provides a code fragment and 4 potential solutions in the form of flow diagrams, one of which represents the code’s logic as a flow diagram. In question 3 this is reversed, the stem is a structure diagram and the 4 possible answers are code fragments.

Change of logic (CL) problems are defined as questions where the student is given a code fragment and the solution is a code fragment that should give the same result but the logic of the algorithm has been altered (or reversed). Question 4 is a CL problem.

At first glance (Table 2) our students do not appear to show such a marked difference in the frequency of doodling between FC and SC type questions as was observed by the Leeds group. For FC questions we found a range of 47-73% annotations while the Leeds group saw a range of 66-88% (mean = 57%, (77%)). For SC questions we found 31-50% of students used annotations while in the Leeds study they saw a range of 38-55% (mean = 41%, (41%)). So while our results still agree with the Leeds study in that FC questions resulted in more doodles than SC type questions we did not see such a marked divide.

CL questions appear to encourage doodling at about the same level as SC type questions although, because of the nature of the CL question different doodle types were observed. In particular a new doodle was observed, the range doodle (Figure 1).

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>52</td>
<td>7</td>
<td>24</td>
<td>31</td>
<td>37</td>
<td>34</td>
<td>36</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td>%</td>
<td>73 (88)</td>
<td>10</td>
<td>33</td>
<td>43</td>
<td>51</td>
<td>47</td>
<td>50</td>
<td>31</td>
<td>56 (79)</td>
</tr>
<tr>
<td>type</td>
<td>FC</td>
<td>CR</td>
<td>CR</td>
<td>CL</td>
<td>FC</td>
<td>FC</td>
<td>SC</td>
<td>SC</td>
<td>FC</td>
</tr>
</tbody>
</table>

Table 2: Number and percentage of annotated questions. The number in parentheses is the percentage reported by McCartney et al. (2004) for the same question.

CR type questions did not encourage doodling (10-33% annotated, %mean = 22) and indeed did not encourage tracing. Of the students who doodled on questions 2 and 3 the main doodle types observed were X, U and A that are all categorised as low ranking doodles. These doodle types are recognised strategies for arriving at an answer to MCQs in any discipline.
The students doodled 3 times more frequently on question 3 than on question 2 although the questions were similar. The most likely reason is because question 3 was the one they found to be more difficult question (question 3 was the 5th hardest while question 2 was the easiest).

<table>
<thead>
<tr>
<th>Type</th>
<th>Hi S, T &amp; O</th>
<th>Med K, N, P, R &amp; C</th>
<th>Low A, U &amp; X</th>
<th>Blank B</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC</td>
<td>29 (81)</td>
<td>25 (72)</td>
<td>4 (11)</td>
<td>42 (120)</td>
</tr>
<tr>
<td>SC</td>
<td>13 (18)</td>
<td>15 (22)</td>
<td>12 (17)</td>
<td>60 (85)</td>
</tr>
<tr>
<td>CR</td>
<td>3 (4)</td>
<td>1 (1)</td>
<td>18 (25)</td>
<td>79 (112)</td>
</tr>
<tr>
<td>CL</td>
<td>0 (0)</td>
<td>28 (20)</td>
<td>15 (11)</td>
<td>56 (40)</td>
</tr>
</tbody>
</table>

Table 3: Percentages of annotations for each question type. Numbers in parentheses present the counts.

Table 3 shows that certain types of questions are more likely to elicit certain doodle types. FC questions elicited more high level and medium level doodles than any other type of question. While FC, CL and SC questions encouraged students to use doodles that have some level of programming strategy, CR questions resulted in MCQ low ranking doodles that were a result of the MCQ question format. The low ranking doodles do not directly contribute to the students understanding of the code itself.

5.4 Reasoning vs. tendency to doodle

The tenth question in the problem set was a short answer question. This question was analysed (for a full description see Whalley et al. (2006) and Lister et al. (2006)) using a set of categories (Table 4) based on the SOLO taxonomy (Biggs and Collis 1982). This taxonomy allowed us to arrive at a level of reasoning for each student. Using this reasoning classification for each student that attempted question 10 we were able to investigate whether or not a student’s ability to reason has some correlation with their tendency to doodle.

Our SOLO classification of students answers to question 10 found that there were 22 relational (R), 28 multistructural (M) and 13 unistructural (U). Where no answer was provided and where a prestructural response was given the students were omitted from this section of the analysis. There was no difference found between the level of reasoning of the students, namely R, M and U, and the probability that they would use doodles to help obtain an answer (F = 0.03239, p = 0.725) in the 9 MCQs. This was unexpected. It had been assumed that multistructural and unistructural students would have a stronger tendency to doodle than relational students. Multistructural and unistructural students in general are unable to see the overall purpose of the code and analyse a code fragment at best line by line. We had expected that they would have needed to use high and medium level doodles to arrive at an answer.

**Table 4: SOLO categories for question 10**

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>80</td>
<td>15</td>
<td>30</td>
<td>40</td>
<td>60</td>
<td>50</td>
<td>60</td>
<td>25</td>
<td>55</td>
</tr>
<tr>
<td>Others</td>
<td>71</td>
<td>9</td>
<td>36</td>
<td>51</td>
<td>53</td>
<td>49</td>
<td>51</td>
<td>29</td>
<td>58</td>
</tr>
</tbody>
</table>

Table 5: Comparison of tendency to doodle by relational and non-relational students

On question 7, the most difficult question, the relational students did indeed doodle more. However, on question 8 the second most difficult question, the non-relational students doodled more. This supports the first finding that there was no difference between the level of reasoning that a student used in a short answer question and the probability that they would use doodles to help solve an MCQ. Furthermore we can conclude that in this survey there was no relation between the difficulty of the question, the relational thinking level as classified on question 10 and the use of doodles.

Davis (1993) found that experts spend more time annotating than novices. If, as we argue, relational students have a higher level of expertise than the other students then we would expect them to doodle more and especially to use tracing more as a strategy for problem solving. However our findings show that the higher-level thinking students did not adopt this technique any more than other students.
Because we had only one question to arrive at a SOLO classification for each student we cannot draw a strong conclusion from these findings. Further investigation is required where questions are designed to elicit the SOLO classification for each student.

6 Conclusion

Our study extended the Leeds study by adding a short answer question (question 10) to the research instrument. This question allowed us to use a recognised cognitive framework to determine the level of reasoning of each student. We found that while most of the students worked at the multistructural level a subset functioned at the relational level, a level of thought that we recognise as being similar to the level an expert would reach when solving a problem. With students assigned a level of reasoning we were able to investigate whether or not the way a student approached problem solving affected their use of doodles. We found that there seemed to be no relation between the level of reasoning and the use of annotation by novice student programmers.

This finding was surprising. We expected that M students would find that they needed to use doodles to solve any but the simplest problems and that the higher-level thinkers would have started to use tracing as a strategy when faced with a difficult problem.

We have confirmed many of the findings of the Leeds working group. Our number of students was larger than the number investigated by the Leeds group and therefore provides weight to their findings. By duplicating two of their questions and designing new questions we have been able to take a fresh look at many of their conclusions. Indeed our problem set not only contained different questions but also identified two new question types. Even though our question set was more diverse in question type we confirmed three key findings of the Leeds group:

1. Doodles: code annotations on paper tend to help students arrive at the correct answer.
2. Tracing: students are more likely to trace on fixed code questions than skeleton code questions.
3. Higher achievers are more likely to doodle.

Additionally, we found that certain types of questions elicit certain types of doodles. In particular certain types of questions encourage students to use doodles that are relevant to understanding the code whereas other types or questions do not.

Although we grouped our questions by type and rank it might be argued that certain doodles actually arise because of the constructs in the code rather than the question type. For example it could be argued that tracing would be a more appropriate strategy when the code fragment contained a loop.

Finally, the analysis in this paper highlights a number of interesting questions:

1. Why in the majority of circumstances do students not annotate their scripts?
2. When students do doodle what motivates them to do so?
3. When students do annotate are they using doodles that are appropriate for the question type?

7 References


conference on Innovation and Technology in Computer Science Education, Bologna, Italy. 118-122, ACM Press.


---

**Appendix**

The questions have been reformatted to fit into the journal layout.

**Question 1 (Question 5: Leeds).**

Consider the following code fragment:

```java
int[] x = {0, 1, 2, 3};
int temp;
int i = 0;
int j = x.length-1;
while (i < j){
    temp = x[i];
    x[i] = x[j];
    x[j] = 2*temp;
    i++;
    j--;
}
```

After this code is executed, array “x” contains the values:

a)  {3, 2, 2, 0}  
b)  {0, 1, 2, 3}  
c)  {3, 2, 1, 0}  
d)  {0, 2, 4, 6}  
e)  {6, 4, 2, 0}

**Question 9 (Question 2: Leeds).**

Consider the following code fragment:

```java
int[] x1 = {1, 2, 4, 7};
int[] x2 = {1, 2, 5, 7};
int i1 = x1.length-1;
int i2 = x2.length-1;
int count = 0;
while ((i1 > 0) && (i2 > 0)){
    if (x1[i1] == x2[i2]){  
        ++count;
        --i1;
        --i2;
    } else if (x1[i1] < x2[i2]){  
        --i2;
    } else  
        // x1[i1] > x2[i2]  
        --i1;
}
```

After the above while loop finishes, “count” contains what value?

a)   3  
b)   2  
c)   1  
d)   0
**Question 4: a CL question**

*A change of logic question where the Boolean logic is reversed but the code purpose remains the same.*

This segment of code checks to see if a number is within a given range. Note:

- \(i\text{MIN\_NUM}\) and \(i\text{MAX\_NUM}\) are both constant integers, where \(i\text{MAX\_NUM} > i\text{MIN\_NUM}\).
- \(i\text{Num}\) is an integer variable;
- \(b\text{Valid}\) is a Boolean variable.

```c
if ((iNum<i\text{MIN\_NUM}) || (iNum>i\text{MAX\_NUM})){
    bValid = False;
}else{
    bValid = True;
}
```

Which of the following pieces of code will give exactly the same result as the code above?

a)  
```c
if ((iNum>=i\text{MIN\_NUM}) && (iNum<=i\text{MAX\_NUM})) {
    bValid = True;
} else {
    bValid = False;
}
```

b)  
```c
if((iNum>i\text{MIN\_NUM}) && (iNum<i\text{MAX\_NUM})){
    bValid = True;
}else{
    bValid = False;
}
```

c)  
```c
if ((iNum>i\text{MIN\_NUM}) || (iNum<i\text{MAX\_NUM})){
    bValid = False;
}else{
    bValid = True;
}
```

d)  
```c
if ((iNum<=i\text{MIN\_NUM}) || (iNum>=i\text{MAX\_NUM})){
    bValid = False;
}else{
    bValid = True;
}
```

**Question 3: a CR question**

Study the following structure diagram:

Array1 and Array2 are both arrays containing \(i\text{MAX}\) integers. Which of these code segments correctly implements the logic shown in the above diagram?

a)  
```c
iIndex1 = 0;
iIndex2 = i\text{MAX};
while (iIndex1 <= i\text{MAX}) {
    iIndex2--;
iArray1[iIndex1] = iArray2[iIndex2];
iIndex1++;
}
```

b)  
```c
iIndex1 = 0;
iIndex2 = i\text{MAX};
while (iIndex1 < i\text{MAX}) {
    iIndex2--;
iArray1[iIndex1] = iArray2[iIndex2];
iIndex1++;
}
```

c)  
```c
iIndex1 = 0;
iIndex2 = i\text{MAX};
while (iIndex1 <= i\text{MAX}) {
    iIndex2--;
iArray2[iIndex2] = iArray1[iIndex1];
iIndex1++;
}
```

d)  
```c
iIndex1 = 0;
iIndex2 = i\text{MAX};
while (iIndex1 < i\text{MAX}) {
    iIndex2--;
iArray2[iIndex2] = iArray1[iIndex1];
iIndex1++;
}
```

For the full problem set please see the BRACElet project website at [http://online.aut.ac.nz/Bracelet/repository.nsf/](http://online.aut.ac.nz/Bracelet/repository.nsf/)