On the Evidence for a Learning Hierarchy in Data Structures Exams

Thomas Pelchen
University of Technology Sydney
Sydney, Australia
Thomas.Pelchen@student.uts.edu.au

Luke Mathieson
University of Technology Sydney
Sydney, Australia
Luke.Mathieson@uts.edu.au

Raymond Lister
University of Technology Sydney
Sydney, Australia
Raymond.Lister@uts.edu.au

ABSTRACT

Several previous research studies have found a relationship between the ability of novices to trace and explain code, and the ability to write code. Harrington and Cheng refer to that relationship as the Learning Hierarchy. However, almost all of those studies examined students at the end of their first semester of learning to program (i.e. CS1). This paper is only the third paper to describe a study of explain in plain English questions on students at the end of an introductory data structures course. The preceding two papers reached contradictory conclusions. Corney et al. presented results consistent with the Learning Hierarchy identified in the CS1 studies. However, Harrington and Cheng presented results for data structures students suggesting that the hierarchy reversed by the time students had progressed to the level of learning about data structures; that is, tracing and explaining were skills that followed writing. In our study of data structures students, we present results that are consistent with the Learning Hierarchy derived from the CS1 students. We believe that the reversal identified by Harrington and Cheng can occur, but only as a consequence of a mismatch in the relative difficulty of tracing, explaining and writing questions.

CCS CONCEPTS

• **Social and professional topics → Computer science education**.

KEYWORDS

programming, data structures, explain in plain English

1 INTRODUCTION

Lopez et al. [7] developed a model for the relationship between the ability of novice CS1 programmers to read, explain, and write code, using data obtained from an end-of-semester exam. The upper portion of the model is shown in Figure 1. The R^2 values in Figure 1 indicate that student scores on tracing questions alone accounted for only 15% of the variation in student scores on writing questions, and student scores on explaining questions alone accounted for only 7% of the variation in student scores on writing questions. However, the combination of scores on tracing and explaining accounted for 46% of the variation on writing questions.

A common, but not universally accepted, interpretation of the Lopez et al. model is that a student's ability to trace small pieces of code precedes their ability to explain small pieces of code and the combination of tracing and explaining precedes a student's ability to systematically write small pieces of code. Harrington and Cheng [4] refer to this model as the Learning Hierarchy. Work subsequent to Lopez et al. has provided further empirical support for this Learning Hierarchy [6, 8, 11]. Venables et al. [11] reported that for their data the combination of scores on tracing and explaining accounted for 66% of the variation in students scores on writing questions. A theoretical explanation has been proposed for these results, using neo-Piagetian theory [5, 10], and most recently methods of instruction for introductory programming have been proposed in which code tracing and explaining are taught as part of teaching code writing [9, 12].

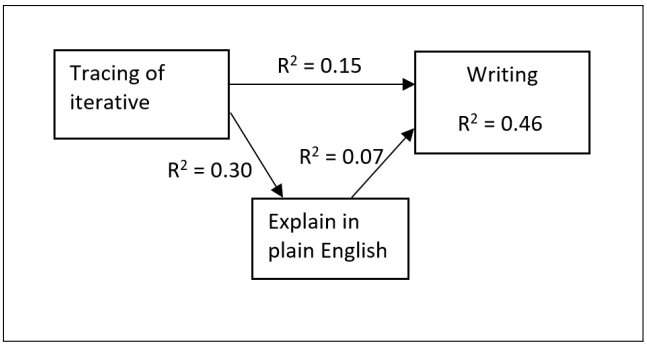

Figure 1: The upper portion of the Lopez et al. model

While there have been several studies of the Learning Hierarchy in CS1 students, there have been few investigations of a possible hierarchy for programming students further into their degree; specifically, there have been just two studies of students studying data structures and algorithms. One of those studies [2] produced evidence consistent with the same Learning Hierarchy (i.e. tracing, explaining and writing). However the other study produced some evidence that led the researchers, Harrington and Cheng [4], to conclude that the Learning Hierarchy had ceased to apply by the time students had reached a data structures course. Furthermore, among students who did show a difference in their scores on code tracing questions and code writing questions, the higher score was as likely to be among the tracing questions as among the writing questions, suggesting that the ability to trace data structures-related code did not precede the ability to write such code. Thus, in this paper, the primary research question is:

- **RQ1:** Can we reconcile the difference between the Harrington and Cheng study [4] with the other study of a data structures exam [2] and the studies of CS1 exams?

Our principal method for exploring that question is to analyze data from a data structures exam conducted at the authors' institution. Our exam did not require students to write code, but it did require students to both trace code and explain code. Thus our data cannot be used to directly explore what skills precede code writing, but our data does allow us to explore the quantitative methods used to establish a relationship between any two skills in the hierarchy, since the methods used to establish a relationship between the combination of tracing and explaining to writing are the same as the methods used in this paper to study the relationship between tracing and explaining.

Our data also allows us to explore another research question:

- **RQ2:** Are there types of questions in addition to tracing questions that show a statistical relationship with explaining code?

1.1 Our Students and their Context

The data structures and algorithms course at the authors' institution covers lists, queues, stacks, hashing and maps, trees, graphs, sorting, string matching and algorithmic theory including greedy algorithms, dynamic programming, divide and conquer, algorithmic analysis and simple complexity theory. It also introduces the students to C++. The cohort consists of 278 students who attempted the final exam. Most of the students (>90%) are majoring in some form of software development. Two programming courses precede this data structures course; both taught in Java.

2 OUR FINAL EXAM

The two-hour open-book exam at the authors' institution contained 20 multiple choice questions (MCQ) and 5 explain in Plain English (EiPE) questions. The remaining questions did not involve tracing code, explaining code, or writing code and are therefore not described in this paper.

Multiple Choice Question 1
What will the following code snippet print out?

```
int i = 5;
int * p = &i;
*p += 10;
cout << p;
```

A. The address of variable i.
B. 15
C. 10
D. An unknown value from memory

Figure 2: Multiple choice question 1, a code tracing question.

Multiple Choice Question 5
Consider the following code:

```
class node {

  private:
    int data;
    node* next;

  public:
    node(int data, node* next){
         this->data = data;
         this->next = next;
    }

    int get_data() const {
         return this->data; }

    bool p() const {
         if (next == nullptr)
            return true;
         if (data < next->get_data())
            return false;
         return next->p();
    }
};
```

and the following code snippet:

```
node* x = new node(1, nullptr);
node* head = new node(2, x);
head->p();
```

What is the result of the code snippet?

A. p is executed on both nodes and both return false.
B. p is executed on both nodes and both return true.
C. p is executed on only the first node and it returns false.
D. p is executed on only the first node and it returns true.

Figure 3: Multiple choice question 5, a tracing question.

Multiple Choice Question 12
Which one of the following statements is TRUE?

A. $15n^2 + 3n + 2 \in O(n\ log_2 n)$
B. $15n^2 + 3n + 2 \in \Omega(n^4)$
C. $15n^2 + 3n + 2 \in O(n^3)$
D. $15n^2 + 3n + 2 \in O(15)$

Figure 4: Multiple choice question 12, a "Big O" question.

Multiple Choice Question 14
Which one of the following statements is FALSE?

A. $n^5 + 2n^4 - 2n + 2 \in O(n^6)$
B. $n^5 + 2n^4 - 2n + 2 \in \Omega(n^4)$
C. $n^5 + 2n^4 - 2n + 2 \in \Theta(n^5)$
D. $n^5 + 2n^4 - 2n + 2 \in \Theta(n^7)$

Figure 5: Multiple choice question 14, a "Big O" question.

Multiple Choice Question 17
Assuming there is no collision, what is the best expected asymptotic (big-oh) running time of inserting a new element into a hashmap which already has n elements?

A. $O(1)$
B. $O(n)$
C. $O(n^2)$
D. $O(log_2 n)$

Figure 6: Multiple choice question 17, a "no category" question.

2.1 The Multiple Choice Exam Questions

The 20 multiple choice questions consisted of 5 code tracing questions, 5 algorithm tracing questions (e.g. in what order would the nodes of the following tree be visited in a preorder traversal?), 4 computational complexity ("Big O") questions, 4 simple recall questions, and two questions not categorized.

The five multiple choice questions that will figure prominently in the following analysis are shown in Figures 2, 3, 4, 5 and 6. The question in Figure 3 is almost the same as a question from Corney et al. [2], while the other MCQs are unique to this study.

2.2 Explain in Plain English (EiPE) Questions

Prior to the exam, students had limited exposure to explain in plain English (EiPE) questions. In the final lecture before the exam, students were advised that there would be EiPE questions in the exam. They were shown some examples of such questions, with suitable answers. Furthermore, in the exam itself, students were given a preamble to the EiPE questions, shown in Figure 7.

3 METHOD

EiPE questions were marked as being either correct or incorrect. Example correct answers for each EiPE question will be given in the results section below.

3.1 Contingency Tables

As both the multiple choice and EiPE questions are graded as being either right or wrong, the combined performance on a given MCQ and and a given EiPE question can be described in a 2×2 contingency table as illustrated in Table 1, where the numbers a, b, c, and d represent the number of students who fell into each of the four

This question is a short answer *explain in plain English* question. You will be given some code and you are required to describe the purpose of it. For example, given the following code:

```
if (a < b)
        std::cout << a << std::endl;
else
        std::cout << b << std::endl;
```

a good answer would be *It prints out the smaller of the two values held by variables a and b.*

Do not give a line by line description of the code. The code should also not be evaluated on whether it compiles or not, your task is to determine the intent of the code, not the ability of the "programmer".

Figure 7: Preamble to Explain in Plain English Questions.

possible categories. A chi-squared test was generally used to assess the statistical significance in such contingency tables. When an individual expected count value was less than 5, a two-tailed Fisher Exact test was used.

The relationship between two questions was considered to be statistically significant when $p < 0.05$. Despite $p < 0.05$ being the traditional threshold for significance, the odds in favour of a p-value just under that threshold being a real effect are about 3:1 [1], and not the 19 to 1 of statistical folk wisdom. A stronger threshold for significance is $p < 0.01$ [1], when the odds of being a real effect are about 15:1 [1]. The traditional $p < 0.05$ was adopted so that more MCQs would come in for consideration, but the reader should be cautious of any p-value such that $0.01 < p < 0.05$

Table 1: A contingency table for two questions

	MCQ Incorrect	MCQ Correct
EiPE Incorrect	a	b
EiPE Correct	c	d

While 278 students attempted the final exam, not all students attempted all EiPE questions. Such non-answers were omitted from our analysis, so some of the contingency tables presented in the results will show a total of less than 278.

3.2 Statistical Measures

An obvious way to consider the combined performance on two exam questions which are both marked as right or wrong is with the phi coefficient of correlation, a measure of how often students answered both questions right or both questions wrong. In terms of the variables a, b, c and d in Table 1, the phi coefficient is:

$$\phi = \frac{(a \times d) - (b \times c)}{\sqrt{a \times b \times c \times d}} \quad (1)$$

The phi coefficient does not describe well an asymmetric relationship between a MCQ and a EiPE question. For example, suppose the MCQ is hard, so that getting the MCQ right may be a good indication that a student will also get the EiPE question right, but getting the MCQ wrong is not a good indicator of whether a student will get the EiPE right or wrong. Such a relationship is better captured by the *likelihood of sufficiency* (*LS*) [3]. If $P(X|Y)$ is the conditional probability of X given Y, then:

$$LS = \frac{P(MCQ\ right \mid EiPE\ right)}{P(MCQ\ right \mid EiPE\ wrong)} = \frac{d \times (a+b)}{b \times (c+d)} \tag{2}$$

The likelihood of sufficiency is most easily understood when its effect is described in terms of odds rather than probabilities. The relationship between odds and probabilities is:

$$odds = \frac{probability}{(1 - probability)} \tag{3}$$

If a probability value is zero then the equivalent odds is also zero. As a probability approaches 1 the equivalent odds approaches infinity. An LS value of 1 means that getting the MCQ right does not affect the odds of getting the EiPE right. An LS value of 2 means that getting the MCQ right doubles the odds of getting the EiPE right.

Conversely, consider an easy MCQ, so that getting the MCQ wrong may be a good indication that a student will get the EiPE question wrong as well, but getting the MCQ right is not a good indicator of whether a student will get the EiPE right or wrong. Such a relationship is captured by the *likelihood of necessity* (*LN*):

$$LN = \frac{P(MCQ\ wrong \mid EiPE\ right)}{P(MCQ\ wrong \mid EiPE\ wrong)} = \frac{c \times (a+b)}{a \times (c+d)} \tag{4}$$

An LN value of 1 means that getting the MCQ wrong does not affect the chances of getting the EiPE right. An LN value of 0.5 means that getting the MCQ wrong halves the odds of getting the EiPE right.

3.3 Plotting (LS, LN) Values

For a given EiPE, the LS and LN numbers for each MCQ can be plotted on a Cartesian plane, as shown in Figure 8. The x-axis corresponds to the LS values and the y-axis corresponds to the LN values. For an MCQ question with a statistically significant relationship to the EIPE, if the associated (LS,LN) coordinate pair fall into any of the shaded regions of Figure 8, the relationship between that MCQ and EIPE can be described thus:

Region A, "neutral": Whether a student gets the MCQ right or wrong provides no information as to how the student performed on the EiPE.

Region B, "easier": The MCQ is so easy it is unlikely that a student who gets such a question wrong will get the EiPE right. However, if a student gets this MCQ right, it provides no information as to how the student performed on the EiPE.

Region C, "harder": The MCQ was so hard it is unlikely that a student who gets such a question right will get the EiPE wrong. However, if a student gets this MCQ wrong, it provides no information as to how the student performed on the EiPE.

Region D, "correlated": While student performance on the MCQ and the EIPE correlate, the relatively low LS value and relatively high LN value indicate there is little or no overlap between the knowledge and skills required for the MCQ and the EIPE. To express that another way: high performing students tend to answer hard questions correctly, irrespective of the question content, while low performing students tend to get those hard questions wrong.

Region E, "prerequisite": The LS value is high enough and the LN value low enough so that the MCQ requires skills or knowledge of the student that are prerequisite for answering the EIPE correctly.

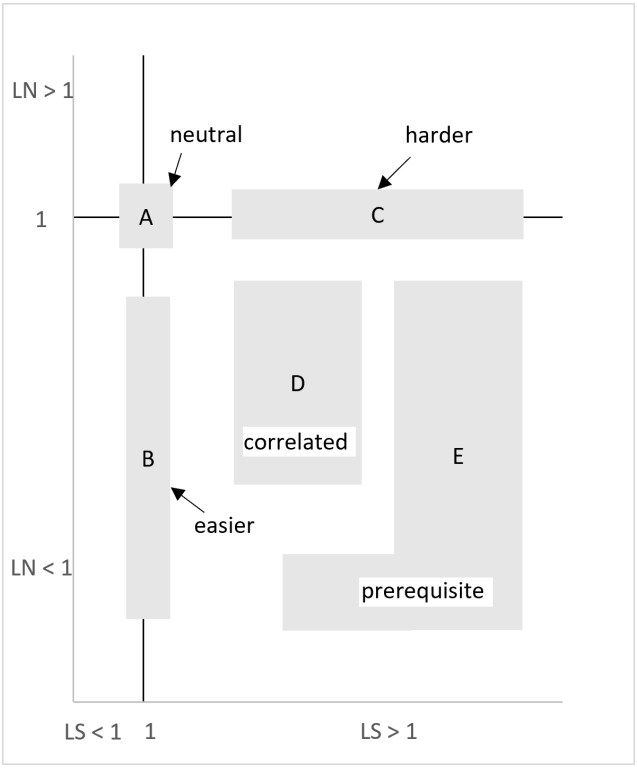

Figure 8: A categorization of how MCQ questions relate to an EiPE question in a plot of (LS,LN) values

For simplicity in introducing the regions in Figure 8, the regions have been drawn as distinct, whereas in reality the regions overlap.

Note that the categorization of MCQs into five regions of Figure 8 only applies to MCQs that have a statistically significant relationship with the EiPE.

Suppose two exam questions, Qx and Qy, have a statistical significant relationship. If getting Qx right significantly reduced the chances of getting Qy right, then Qx or Qy would be a perverse exam question. The same applies if getting Qx wrong significantly

improved the chances of getting Qy right. It follows that non-perverse, statistically significant (LS,LN) data points in plots like Figure 8 appear in the lower right quadrant, where LS>1 and LN<1. As will be seen in plots of real data in the following "results" section, sometimes (LS,LN) data points do appear outside that lower right quadrant, but only for MCQs that do not have a statistically significant relationship with the EiPE.

4 RESULTS

In this section, results are presented from 3 of the 5 EiPE questions in the exam, including the hardest question and the easiest question.

4.1 Q22i: Check for a Sorted Linked List

The code for Q22i is shown in Figure 9. A correct answer is "it checks if the data values in the linked list are in descending order". This question is similar to a question from Corney et al. [2]. For an answer to be correct, a student had to be explicit about the descending order. Only 25% of the class answered this question correctly. Of the 5 EiPE questions in the exam, this was the hardest.

As indicated in Figure 9, the code for this EiPE question was also used in the MCQ5 tracing question. The exam paper did not explicitly point out to students that the same code was used in Q22i and MCQ5. It is unknown how many students did notice that the code was the same and were able to take advantage of their trace for MCQ5 to answer Q22i.

Question 22i
Consider the following code:

... In the actual exam, the code for class "node" from MCQ5 (i.e. Figure 3) was included here in Q22i, except for the "code snippet" of MCQ5 ...

Assume you have a linked list constructed of such nodes. What property of the list does calling the function p on the head of the list test?

Figure 9: Explain in plain English question Q22i.

Table 2 shows statistics for the three statistically significant MCQs for Q22i ($p < 0.05$). Only 44% of the class answered MCQ5 correctly, making it the hardest of the 20 MCQs. (Hence the row beginning with "Rank" in Table 2 shows 20 for MCQ5.) The phi correlation coefficient and LS value for Q22i and MCQ5 are the highest for any combination of the 20 MCQs and five EiPE questions. Those high values might be expected, given that MCQ5 and Q22i share common code.

Tables 3, 4, and 5 show the contingency tables for the relationship between Q22i and MCQ5, MCQ12 and MCQ14 respectively. While all three MCQs are significant at the $p < 0.05$ level, note that only the MCQ5 tracing question is significant at the $p < 0.01$ level.

For MCQ5, the data is consistent with a learning hierarchy where tracing skill precedes explaining skill, not the reverse found by Harrington and Chen in their study. That is, in Table 3, the number of students who answered the MCQ tracing question incorrectly but the EiPE correctly (i.e. 13 in Table 3) is far less than the number of students who answered the MCQ tracing question correctly but the EiPE incorrectly (i.e. 70 in Table 3).

Table 2: The three statistically significant MCQs for Q22i

	MCQ5	MCQ12	MCQ14
Percent right	44%	75%	64%
Rank	20 (hardest)	18	19
Significance	p <0.0001***	p = 0.02*	p = 0.03*
ϕ	0.34	0.15	0.14
LS	2.04	1.2	1.25
LN	0.35	0.44	0.61

Table 3: The contingency table for Q22i and MCQ5 (ϕ = 0.34, LS = 2.04, LN = 0.35, p < 0.0001*)**

Q22i & Q5	MCQ Incorrect	MCQ Correct
EiPE Incorrect	112	70
EiPE Correct	13	47

Table 4: The contingency table for Q22i and MCQ12 (ϕ = 0.15, LS = 1.2, LN = 0.44, p < 0.02*)

Q22i & Q12	MCQ Incorrect	MCQ Correct
EiPE Incorrect	48	134
EiPE Correct	7	53

Table 5: The contingency table for Q22i and MCQ14 (ϕ = 0.14, LS = 1.25, LN = 0.61, p < 0.03*)

Q22i & Q14	MCQ Incorrect	MCQ Correct
EiPE Incorrect	70	112
EiPE Correct	14	46

Figure 10 plots the (LS,LN) values for all 20 MCQs, for Q22i. In the figure, only the three MCQs that have a statistically significant relationship ($p < 0.05$) with Q22i are labelled — MCQs 5, 12 and 14. As the key in Figure 11 indicates, MCQ5 is a code tracing question while the other two are "Big O" questions.

In Figure 10, and also later plots of (LS,LN) values, the two axes has been scaled so that the horizontal distance on the printed page from (LS=1, LN=1) to (LS=2, LN=1) is the same as the vertical distance from (LS=1, LN=1) to (LS=1, LN=0.5). The authors believe that this scaling has the intuitive visual effect of giving LS and LN values equal importance.

In Figure 10, and also later plots of (LS,LN) values, the vertical axis has been compressed for values of LN>1. The compression was done to conserve space. The highest LN value the authors saw in all our exam data is shown in this plot, with LN approximately equal to 1.8. As can be seen in the figure, the LS value associated with that LN $\approx$ 1.8 is only slightly less than one and that data point is not statistically significant (since the data point is not labelled).

Summary: On the basis of Figure 10 and its associated data, the authors conclude that (1) "Big O" MCQs 12 and 14 are correlated with this EIPE, but (2) only MCQ5 requires prerequisite knowledge or skills for this EIPE, which is not surprising, given that this MCQ and EIPE share most of the same code.

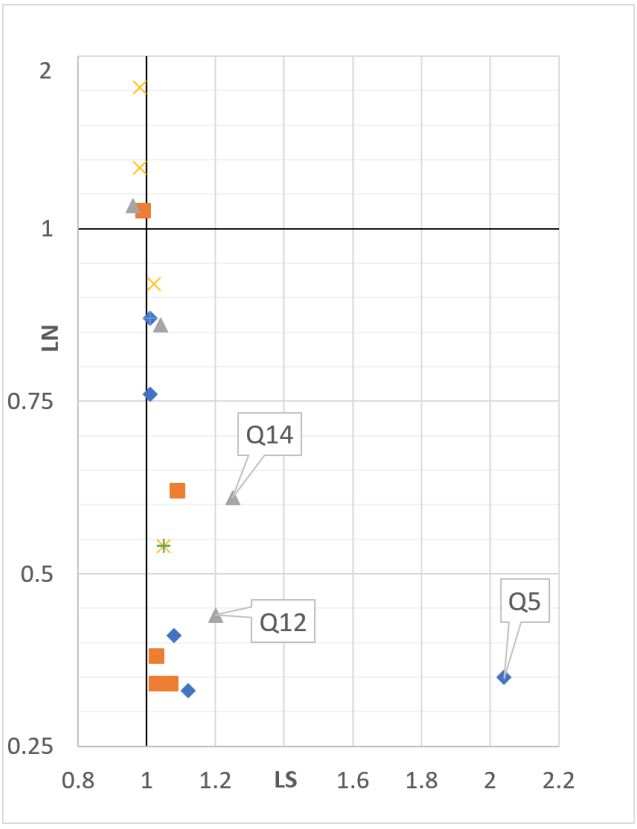

Figure 10: Q22i "Check for a Sorted Linked List " (LS,LN)

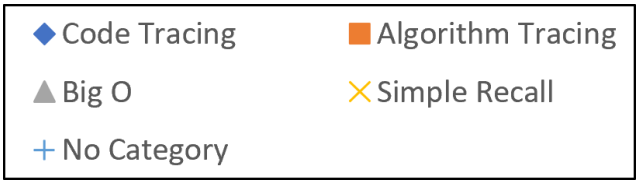

Figure 11: The key for the categorization of MCQ questions in Figure 10 and the other plots of (LS,LN) values

4.2 Q22ii: Sum Contents of a Vector

The code for Q22ii is shown in Figure 12. A correct answer is "it sums the contents of a vector". 68% of the class answered this question correctly. A question like this was used in the original Lopez et al. study of CS1 students [7].

Only two MCQs showed a statistically significant relationship ($p < 0.05$) to this EiPE question, MCQ5 (see Figure 3) and MCQ12 (see Figure 4). The contingency tables for these two MCQs and this EiPE question are shown in Tables 6 and 7.

Table 6, is what we the authors of this paper refer to as being an anomalous contingency table. That is, this table supports the reverse hierarchy conclusion of Harrington and Chen, with the number of students who answered MCQ5 incorrectly but EiPE Q22ii correctly (i.e. 76 students) exceeding the number of students

Question 22ii
In one sentence, explain in plain English what the following method does:

```
int w(const std::vector<int>& v) const {
  int n = 0;
  for(std::vector<int>::iterator itr =
      v.begin(); itr < v.end(); ++itr)
  {
      n = n + *itr;
  }
  return n;
}
```

Figure 12: Explain in plain English question Q22ii.

who answered the MCQ correctly but the EiPE incorrectly (i.e. 25 students).

Figure 13 plots the (LS,LN) values for all 20 MCQs, for this EIPE question. In the figure, only the two MCQs that have a statistically significant relationship with Q22ii are labelled. One of those labelled questions is MCQ5; note that the LS number for MCQ5 is greater than 1, and the LN number is less than 1, even though the contingency table (i.e. Table 6) is anomalous.

Table 6: Anomalous contingency table for Q22ii and MCQ5 (ϕ = 0.28, LS = 2.06, LN = 0.6, p < 0.0001***)

Q22ii & Q5	MCQ Incorrect	MCQ Correct
EiPE Incorrect	67	25
EiPE Correct	76	97

Table 7: The contingency table for Q22ii and MCQ12 (ϕ = 0.18, LS = 1.25, LN = 0.5, p < 0.003**)

Q22ii & Q12	MCQ Incorrect	MCQ Correct
EiPE Incorrect	31	61
EiPE Correct	30	143

The smallest LS value the authors saw in all their exam data is shown in Figure 13 in the coordinate pair (LS≈0.85, LN≈1.6). This data point is also furthest statistically non-significant point, in all the author's exam data, from the LS=1 axis and the LN=1 axis, with LS<1.

Summary: On the basis of Figure 13 and its associated data, the authors conclude that (1) "Big O" MCQ 12 is correlated with this EIPE, but (2) only MCQ5 requires prerequisite knowledge or skills for this EIPE.

4.3 Q22iii: Sorting a Vector

The code for Q22iii is shown in Figure 14. A correct answer is "it sorts the vector". A student did not need to specifically mention

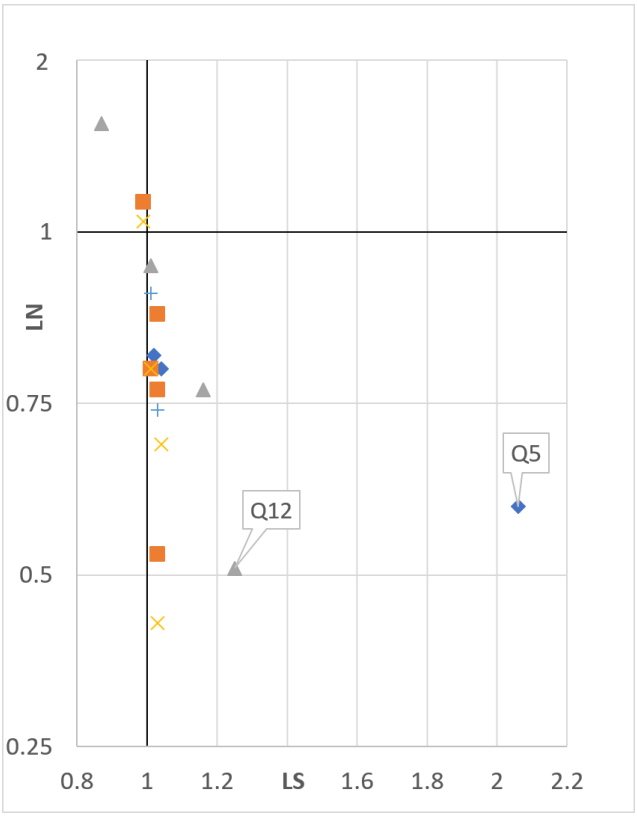

Figure 13: Q22ii "Sum" (LS,LN)

bubblesort. This was the easiest of the five EiPEs with 70% of the class answering correctly.

Five MCQs showed a statistically significant relationship ($p<0.05$) to this EiPE question. The contingency tables for these five MCQs and this EiPE question are shown in Tables 8 to 12. Although MCQ14 is not a tracing question, it's contingency table (i.e. Table 11) is an anomalous contingency table, as the number of students who answered the MCQ incorrectly but the EiPE correctly (i.e. 55 students) just exceeds the number of students who answered the MCQ correctly but the EiPE incorrectly (i.e. 44 students).

Table 8: The contingency table for Q22iii and MCQ1 (ϕ = 0.13, LS = 1.1, LN = 0.48, $p < 0.04^{*}$)

Q22iii & Q1	MCQ Incorrect	MCQ Correct
EiPE Incorrect	13	66
EiPE Correct	14	164

Table 9: The contingency table for Q22iii and MCQ5 (ϕ = 0.21, LS = 1.07, LN = 0.67, $p < 0.0001^{***}$)

Q22iii & Q5	MCQ Incorrect	MCQ Correct
EiPE Incorrect	15	64
EiPE Correct	20	158

Question 22iii
In one sentence, explain in plain English what the following method does:

```
std::vector<int> x(std::vector<int> v){
do {
      bool a = false;
      for (int i=0; i<v.size()-1; i++){
           if (v[i] > v[i+1]){
                int temp = v[i+1];
                v[i+1] = v[i];
                v[i] = temp;
                a = true;
           }
      }
   } while (!a);

   return v;
}
```

Figure 14: Explain in plain English question Q22iii.

Table 10: The contingency table for Q22iii and MCQ12 (ϕ = 0.19, LS = 1.26, LN = 0.49, $p < 0.002^{***}$)

Q22iii & Q12	MCQ Incorrect	MCQ Correct
EiPE Incorrect	27	52
EiPE Correct	30	148

Table 11: The anomalous contingency table for Q22iii and MCQ14 (ϕ = 0.13, LS = 1.24, LN = 0.7, $p < 0.04^{*}$)

Q22iii & Q14	MCQ Incorrect	MCQ Correct
EiPE Incorrect	35	44
EiPE Correct	55	123

Table 12: The contingency table for Q22iii and MCQ17 (ϕ = 0.15, LS = 1.11, LN = 0.41, $p < 0.02^{*}$)

Q22iii & Q17	MCQ Incorrect	MCQ Correct
EiPE Incorrect	12	67
EiPE Correct	11	167

Figure 15 plots the (LS,LN) values for all 20 MCQs, for Q22iii. In the figure, only the five MCQs that have a statistically significant relationship with Q22iii are labelled. Three of the five labelled MCQs – 5, 12, and 14 – also occurred in Figure 10 and two of those MCQs – 5 and 12 – appeared in Figure 13. In this plot, MCQ5 is not as far to the lower right as in the earlier Figures, which is to be expected, given that the code traced in MCQ5 is different from, and harder than, the code in this EiPE question. Never the less, the data point for MCQ5 is distant from the other statistically significant MCQs. The data points for MCQs 12 and 14 in Figure 15 are roughly in the same position as in Figures 10 and 13.

MCQ17 was also statistically significant for this EiPE. The text of that MCQ is shown in Figure 6.

The final statistically significant MCQ in Figure 15 is MCQ1, although it is only just statistically significant (p = 0.04). The text of that MCQ is shown in Figure 2. It is a tracing question, like MCQ5, but a much easier tracing question.

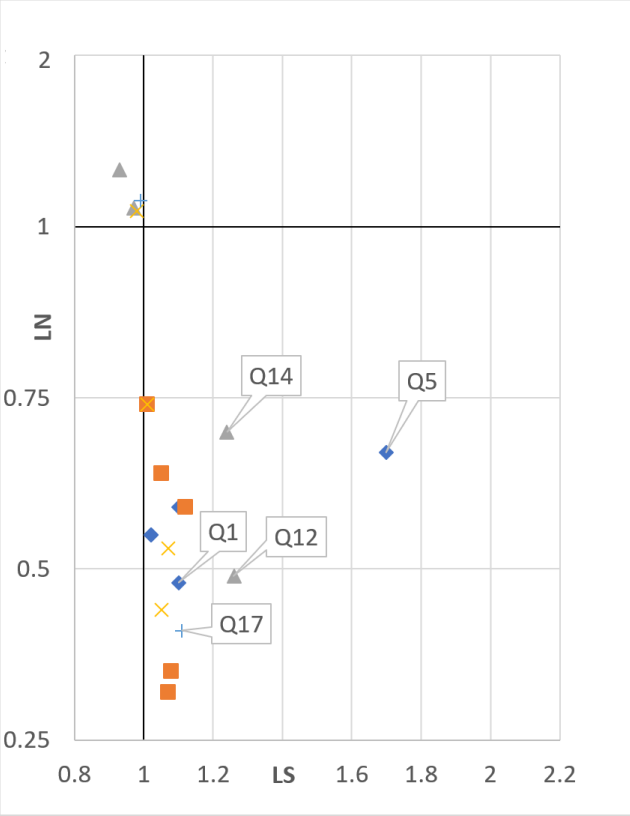

Figure 15: Q22iii "Sorting a Vector" (LS,LN)

Summary: On the basis of Figure 15 and its associated data, the authors conclude that (1) MCQs 1 and 17 are type B "easier" questions for this EiPE, (2) MCQs 12 and 14 are type D questions that correlate with this EIPE, whereas (3) MCQ5 is a type E question, which requires prerequisite knowledge or skills for this EIPE.

5 DISCUSSION

In the previous section, results were presented for 3 of the 5 EiPE questions in the exam. Page limits do not allow for the presentation of all five questions, but the plots of (LS,LN) were similar in all five cases, as follows:

- Tracing question MCQ5 (see Figure 3) was always statistically significant ($p < 0.05$), and was well removed from the other data points, appearing to the lower right, indicating both considerable sufficiency and necessity.
- MCQ 12, the "Big O" question shown in Figure 4 was also always statistically significant ($p < 0.05$).
- A similar "Big O" question, MCQ14 (see Figure 5), was statistically significant for four of the five EiPEs ($p < 0.05$).
- In all cases except for MCQ5, when a question was statistically significant, it appeared in roughly the same position in the plots of (LS,LN). MCQ5 is an exception because it had an especially high LS number for the EiPE question with which it shared code, Q22i.

Also, the easy tracing question MCQ1 shown in Figure 2 was statistically significant for two of the five EiPEs ($p < 0.05$).

As the authors wrote earlier in this paper, some caution should be exercised with the MCQs and EIPEs that exhibit a statistical relationship where $0.01 < p < 0.05$. All the p-values described in this paper are an indication of the likelihood of a successful replication, but only if two important and very restrictive criteria are satisfied: (1) the students in the replication have a similar level of programming skill as the students studied in this paper, and (2) the students in the replication are taught the same way as the students described in this paper. If our same exam questions were used and analyzed at another institution, with different students and different teaching methods, the only relationship we are confident might be reproduced is the relationship between MCQ5 and the EiPEs. That confidence is not just based on the very low p-values for MCQ5. At least just as important are the strong LS and LN numbers for MCQ5.

5.1 Anomalous Contingency Tables

In their data, Harrington and Cheng observed higher writing scores than tracing scores. (See Figure 5 in their paper.) That observation led them to conclude that the unidirectional nature of the hierarchy in CS1 students did not apply by the time students studied data structures.

In this paper, which is a study of the relationship between tracing and explaining, the analogous phenomenon (i.e. explaining scores higher than tracing scores) was sometimes observed in anomalous contingency tables where the number of students who answer an MCQ incorrectly but the EiPE correctly (i.e. cell c in Table 1) exceeds the number of students who answer the MCQ correctly but the EiPE incorrectly (i.e. cell b in Table 1). Of the seven contingency tables that showed a statistical relationship ($p < 0.05$) between any of the five EIPE questions and a tracing question, two tables were anomalous, which is a high ratio that needs explaining if the characterization of "anomalous" is warranted.

One of the anomalous contingency tables involving a tracing question was the table for MCQ5 and Q22ii (see Table 6). MCQ5 is a difficult tracing question, involving recursion. In contrast, EiPE Q22ii is a simple question – simple even for CS1 students – as the code merely sums the integer elements of a vector. That the MCQ is hard and the EiPE is easy has led the authors of this paper to the conjecture that anomalous contingency tables are an indication of a mismatch in difficulty between a tracing question and the explanation question. That conjecture then leads to a guideline when devising questions to be used to study the learning hierarchy: *the code in a tracing question should ideally not contain programming constructs that are not present and considered more advanced than the constructs in the explanation/writing questions.*

The other anomalous contingency table for a tracing question involved an EiPE that is not included in this paper. The EiPE contained code that incremented each each integer in a vector. The

tracing question was the difficult MCQ5 again. Thus this other anomalous contingency table is consistent with our conjecture that anomalous contingency tables are an indication of a mismatch in difficulty between a tracing question and an explanation question.

5.2 Piagetian Theory and Goldilocks Questions

After the early empirical work on a learning hierarchy in CS1 students, a neo-Piagetian theory was developed to explain the developmental stages of the novice programmer [5, 10]. Harrington and Chen's claims about the hierarchy in data structures students are not compatible with neo-Piagetian theory. However, their claims may be compatible with a strict interpretation of classical Piagetian theory. In classical Piagetian theory, once a student has acquired a higher stage of reasoning, the student tends not to regress to a lower stage. That is, in terms of learning to program, while a CS1 student may rely heavily on tracing to understand code, by the time that student has progressed to studying data structures, the student may have acquired the ability to read a piece of code and deduce its purpose, without needing to trace the code. In contrast, according to neo-Piagetian theory, while a data structures student might read code and deduce the purpose of code containing familiar concepts, and not feel the need to trace the code, that same student may regress to a lower form of reasoning (i.e. tracing) as part of learning to accommodate a strange new concept (e.g. recursion).

For empirical studies of the learning hierarchy, neo-Piagetian theory implies the need for "Goldilocks" questions, that are not too hard and not too easy, but are just the right level of novelty and difficulty so that a student will exhibit a regression to tracing code before being able to explain and write similar pieces of code. Thus, not every combination of tracing questions and EIPE questions will generate empirical evidence for a learning hierarchy. Furthermore, if a set of tracing and EiPE questions does generate empirical evidence for a learning hierarchy using one set of students, those same questions may not generate empirical evidence for a learning hierarchy with a set of different students. Prior to the work of Harrington and Chen, researchers studying the hierarchy have relied upon their knowledge of their own students to generate questions with just the right level of novelty and difficulty for their own students. To express this pithily: *Piagetian theory is a theory of development, so empirical studies must use test questions consistent with the students' level of development.*

With the aim of systematically constructing tracing/explain questions of similar difficulty to the writing questions, Harrington and Cheng used a novel and clever approach. They wrote two exam papers, and had half their class do each paper. The code for the tracing question in one paper was a solution for the code writing question in the other exam paper. They argued plausibly that this approach is better than the *ad hoc* approaches used in earlier studies. However, there is an *ad hoc* asymmetry even in this two exam approach – turning a solution for a code writing question into a tracing question requires the addition of input data that needs to be traced, so the difficulty of a tracing question is not just determined by the code, but also by the input data. Perhaps with the implicit recognition of this asymmetry, Harrington and Cheng had their students complete two traces on each piece of code, where the input for one trace was simpler than the input for the second trace. The two tracing problems carried equal weight in the point scoring system. But even with two sets of tracing data, some intuition and perhaps even some trial and error is still required to get the right level of difficulty.

5.3 Tracing and the Full Lopez Hierarchy

Tracing question MCQ5 was a strong predictor for all five EiPEs that the authors studied. In contrast, tracing question MCQ1 (see Figure 2) only appeared in one plot of (LS,LN) values, Figure 15, where it just met the threshold for statistical significance ($p = 0.04$).

The differing importance of MCQ1 and MCQ5 is consistent with the complete Lopez hierarchy [7]. Figure 1 in this paper only shows the upper portion of the Lopez hierarchy. The complete Lopez hierarchy contains two sets of tracing questions. The harder tracing questions, which usually involved some iterative process on an array, are the tracing questions shown in Figure 1. Not shown in Figure 1 is a separate and easier set of tracing questions that appear in the lower portion of the complete Lopez hierarchy.

That MCQ1 is (just) statistically significant for EiPE Q22iii but not statistically significant for the other EiPEs is probably because Q22iii is the easiest of the five EiPEs.

5.4 Tracing, Explaining and Partial Scores

What Harrington and Cheng refer to as their "tracing" question is actually an equally weighted combination of tracing and explanation. As they wrote in their paper:

> ... Students were then asked to trace the output of the function on a simple tree, and then a more complex one. Finally students were asked to provide a sensible name for the function and provide a docstring that clearly explained what the function does. 2 marks were awarded for tracing the simple tree, 2 marks for the complicated tree, and 4 marks for the docstring.

The authors of this paper regard the provision of a sensible function name and a docstring comment as being an exercise in code explanation – arguably a good way of testing code explanation. So the reader should be aware that when Harrington and Chen refer to "tracing", they are referring to a combination of tracing and explaining. We conjecture that a possible weakness of combining tracing and explaining as Harrington and Chen have done is that in the Lopez hierarchy it is the combination of some skill in tracing and some skill in explaining that leads to competence in writing. When the tracing and explaining scores are entwined as in Harrington and Chen's study, it is not clear if a partial score indicates sufficient skill in both tracing and explaining.

Harrington and Cheng did not clarify what type of answers received a partial score. In contrast, most of the earlier work on the learning hierarchy graded tracing and explaining questions in a binary fashion: as being right or wrong. There is no paradigm yet for studying the learning hierarchy, so Harrington and Cheng were not wrong to use partial scores, but perhaps their use of partial scores accounts for some of their differences from other studies of the hierarchy.

5.5 The Big O

MCQ12 and MCQ14 are both "Big O" questions that correlate with several EiPE questions. At least part of the reason why these MCQs correlated with EiPE questions is indicated by Table 2, in the row beginning with "Rank". As shown in the table, these Big O questions are two of the three hardest MCQs in the exam (along with MCQ5). Irrespective of the content of questions, good students tend to get hard questions right, while poor students tend to get hard questions wrong.

As the authors wrote earlier in this paper, the regions in Figure 8 overlap. There is probably no algorithm, using statistics alone, to unambiguously separate type D "correlated" questions from type E "prerequisite" questions. In addition to the statistics, the content of the questions must considered. These two Big O questions do not feature any code, let alone data structures, and so that, in conjunction with the statistics, led the authors to classify these two Big O questions as "correlated".

6 CONCLUSION

With respect to our first research question, the results described in this paper have led us to the conjecture that Harrington and Cheng's reversal effect occurs when there is a mismatch in the relative difficulty of tracing, explaining and writing questions. Of course, this paper is only the third study of the learning hierarchy with respect to students studying data structures, so further studies are warranted.

With respect to our second research question, despite our exam containing several different types of MCQs, we did not identify any new type of MCQ question that appear to require prerequisite knowledge or skills for explaining code – tracing remains the only skill identified thus far that is prerequisite for explaining code.

Given the need for "Goldilocks" questions, some readers may argue that the Learning Hierarchy is a tautology: it exists only for data that confirms its existence. The authors argue that such a position is severe. Even laws of physics, such as Newtonian mechanics, only apply under a given range of circumstances. The numerous published studies that have presented data supporting a hierarchy (albeit that most are for CS1 students) suggest that the range of circumstances under which the hierarchy applies is not overly restricted or quirky.

This paper has been framed within a positivistic framework. That is, the paper is framed as a study of the possible existence of an objectively real hierarchy, in the same way as science studies, for example, Newton's theory of gravitation. However, from a pragmatic pedagogical point of view, it may be more productive to pursue the Learning Hierarchy as if it is a social construction. That is, it may be more productive to think of the hierarchy as a pedagogical framework for teaching programming, where tracing code and reading/explaining code are taught before students begin to write large quantities of code. From that social construction perspective, a good tracing question and a good explanation question are those that are most useful in preparing a student to write code. Perhaps the type of empirical evidence presented in this paper does not demonstrate the existence of the hierarchy in the same way that the orbit of the moon demonstrates the existence of gravity. Instead, perhaps empirical evidence like that presented in this paper demonstrates that suitable tracing, explaining and writing questions have been constructed for a pedagogy in which tracing and reading code are to be taught prior to students writing their own code.

7 ACKNOWLEDGMENT

The authors thank Brian Harrington and Nick Cheng for sharing their exam questions.

REFERENCES

[1] D. Colquhoun. 2017. The Reproducibility Of Research And The Misinterpretation Of P Values. *Royal Society Open Science* 4, 12 (Dec. 2017). http://dx.doi.org/10.1098/rsos.171085

[2] Malcolm Corney, Sue Fitzgerald, Brian Hanks, Raymond Lister, Renee McCauley, and Laurie Murphy. 2014. 'Explain in Plain English' Questions Revisited: Data Structures Problems. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 591–596. https://doi.org/10.1145/2538862.2538911

[3] R. Duda, J. Gaschnig, and P. Hart. 1981. Model Design in the Prospector Consultant System for Mineral Exploration. In *Readings in Artificial Intelligence*, B. L. Webber and N. J. Nilsson (Eds.). Kaufmann, Los Altos, CA, 334–348.

[4] Brian Harrington and Nick Cheng. 2018. Tracing vs. Writing Code: Beyond the Learning Hierarchy. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 423–428. https://doi.org/10.1145/3159450.3159530

[5] Raymond Lister. 2011. Concrete and Other neo-Piagetian Forms of Reasoning in the Novice Programmer. In *Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114 (ACE '11)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 9–18. http://dl.acm.org/citation.cfm?id=2459936.2459938

[6] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship Between Explaining, Tracing and Writing Skills in Introductory Programming. *SIGCSE Bull.* 41, 3 (July 2009), 161–165. https://doi.org/10.1145/1595496.1562930

[7] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships Between Reading, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08)*. ACM, New York, NY, USA, 101–112. https://doi.org/10.1145/1404520.1404531

[8] Laurie Murphy, Sue Fitzgerald, Raymond Lister, and Renée McCauley. 2012. Ability to 'Explain in Plain English' Linked to Proficiency in Computer-based Programming. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research (ICER '12)*. ACM, New York, NY, USA, 111–118. https://doi.org/10.1145/2361276.2361299

[9] Sue Sentance, Jane Waite, and Maria Kallia. 2019. Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education* 29, 2-3 (2019), 136–176. https://doi.org/10.1080/08993408.2019.1608781 arXiv:https://doi.org/10.1080/08993408.2019.1608781

[10] Donna Teague and Raymond Lister. 2014. Longitudinal Think Aloud Study of a Novice Programmer. In *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148 (ACE '14)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 41–50. http://dl.acm.org/citation.cfm?id=2667490.2667495

[11] Anne Venables, Grace Tan, and Raymond Lister. 2009. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop (ICER '09)*. ACM, New York, NY, USA, 117–128. https://doi.org/10.1145/1584322.1584336

[12] Benjamin Xie, Dastyni Loksa, Greg L. Nelson, Matthew J. Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Andrew J. Ko. 2019. A theory of instruction for introductory programming skills. *Computer Science Education* 0, 0 (2019), 1–49. https://doi.org/10.1080/08993408.2019.1565235 arXiv:https://doi.org/10.1080/08993408.2019.1565235