On the Frequency of Words Used in Answers to Explain in Plain English Questions by Novice Programmers

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

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

ABSTRACT

Most previous research studies using Explain in Plain English questions have focussed on categorising the answers of novice programmers according to the SOLO taxonomy, and/or the relationship between explaining code and writing code. In this paper, we study the words used in the explanations of novice programmers. Our data is from twelve Explain in plain English questions presented to over three hundred students in an exam at the end of the students' first semester of programming. For each question, we compare the frequency of certain words used in correct answers, between students who scored a perfect twelve on all the Explain in plain English questions and students with lower scores. We report a number of statistically significant differences in word frequency between the students who answered all questions correctly and students who did not. The students who answered all twelve questions correctly tended to be more precise, more comprehensive, and more likely to choose words not explicitly in the code, but instead words that are an abstraction beyond the code.

CCS CONCEPTS

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

KEYWORDS

Novice Programmers, Explain in Plain English

1 INTRODUCTION

Since Explain in Plain English questions (EiPE) were first presented at the ACE 2006 conference [22], EiPE questions have become a common way to study the code reading and understanding of novice programmers. In most of those studies, the focus has been on comparing the performance of students on EiPE questions and how that relates to the performance of those students on code tracing and code writing questions. In those past studies, the number of EiPE questions used was low, often only two or three EiPE questions, with the largest number of EiPE questions used in a single past study being six [3]. In this paper, we had our student subjects answer a dozen EiPE questions.

Our motivation arose from reading some literature in the area of Linguistic Inquiry and Word Count (LIWC) [8, 16]. In two LIWC studies, where both studies were in knowledge domains that were not computer-related, Kim et al. [7] and Gobbo and Chi [6] found that experts and novices differed in the frequency with which they used certain words. Researchers in LIWC claim that they have identified patterns in the use of words that are domain-independent [7]. If certain patterns of language usage are associated with the degree of expertise, and those patterns are domain-independent, then the exact words used by a student when describing programming code may be an indicator to a teacher of the developmental stage of the student. Inspired by that literature in LIWC, our primary research question was as follows:

> **RQ1:** Among students who answer a specific EiPE question correctly, is there a difference between the answers of students who did well on other EiPE questions and students who did not do as well on other EiPE questions?

Having our students answer twelve EiPE questions also allowed us to ask a secondary research question:

> **RQ2:** Are some EiPE questions harder than others?

Several of our 12 EiPE questions have been used in earlier studies of EiPE questions. This paper is the first time that all of these previously published EiPE questions have been compared on the same student cohort, thus for the first time providing the opportunity to directly compare student performance on all these questions.

1.1 Institutional Context

The study presented in this paper used data collected from over 300 students enrolled in an introductory programming subject at the authors' university. The programming language taught is Java.

An earlier study of EiPE questions provided some evidence that the English-speaking background of students may affect their ability to answer EiPE questions [15]. Direct data about the English-speaking background of the students sampled in this study was not available. However, it is known from regular surveys of this class in other semesters that typically 80% of the class regard English as their first language.

2 METHOD

2.1 The End-of-Semester Exam Paper

The 12 EiPE questions were part of the end-of-semester exam. Every question began with the same preamble:

> In one sentence that you should write in the empty box below, explain in plain English what this code does.

The "empty box below" which followed each question was the full-text width of the single-column page and was 2cm in height. For most students, the box provided ample room for their answer. Every student's full answer was transcribed and included in our analysis, even when it extended beyond the box.

Prior to the exam, students had limited exposure to EiPE questions. The very first EiPE question presented below (Q25) was used on a non-assessable worksheet in the first two weeks of semester. In the final "exam hints" lecture, students were advised that there would be EiPE questions in the exam, and they were shown a small number of examples of such questions, and what constituted a correct answer, using code that had been studied during semester, but none of those examples used code from the actual exam questions.

2.2 Correct, Relational Answers Only

The reader should note that the student answers analyzed in this paper are all correct answers; incorrect answers are ignored. The motivation for doing so is that the exact words a student uses in a correct answer to a single EiPE question may be an indication of how well the student comprehends code in general. Furthermore, for this study, a correct answer must be, in terms of the SOLO taxonomy, a relational answer [2, 10, 14, 22]. A line-by-line multi-structural answer is considered incorrect.

2.3 Data Collection

The handwritten answers from the exam papers of 334 students were transcribed into a digital text format for analysis. Of those 334 students, 31 students scored zero on the 12 EiPE questions. Those 31 students are included in the statistical reporting below.

2.3.1 Transcription Rules. To maintain consistency in the way that the words were transcribed and analyzed, the following transcription rules were applied:

(1) Spelling mistakes were corrected, to ensure that the student's intended word was analyzed.
(2) When a word could not be read, due to poor handwriting, <*illegible*> was entered.
(3) The names of variables were encased with quotation marks, thus distinguishing the variable name from the normal use of the word; for example, a variable called 'number' is distinguished from the common noun *number*.
(4) Similar words, for example, "number", "numbers" and "num", were collapsed into the same word.

2.4 Data Analysis

Students from high, medium and low performance bands were studied:

- **Band 12:** the 36 students who answered all 12 EiPE questions correctly.
- **Band 9:** the 39 students who answered 9 of the 12 EiPE questions correctly. This medium band is roughly equidistant from the other two bands, in terms of a score out of 12. On average only 9/12×39 ≅ 29 of these band 9 students would have answered a question correctly and thus be eligible for inclusion in the analysis of that question.
- **Band 4-7:** the 69 students who answered 4 to 7 of the 12 EiPE questions correctly. Even lower performing students were not used because a useful sample size is needed per question. Consider the students who answered 1, 2 or 3 questions correctly. On average only approximately nine of those students would have answered a specific question correctly and thus be eligible for inclusion in the analysis for that question. In the 4-7 band, an average of approximately 33 students would have answered a specific question correctly.

A word was counted only once per student, as our interest is in a student's lexicon, not how many times they use a word.

The comparison of a word between two performance bands implies a 2×2 contingency table as illustrated in Figure 1. In general, a chi-squared test was used to assess the statistical significance in such contingency tables. However, when an individual expected count value was less than 5, a two-tailed Fisher Exact test was used [12]. Also, when comparing the frequency of use of a particular word in all three performance bands simultaneously (i.e. a 3×2 contingency table), a Fisher-Freeman-Halton Exact test was used when 20% of the expected count values were less than 5 or a single expected count value was less than 1 [12].

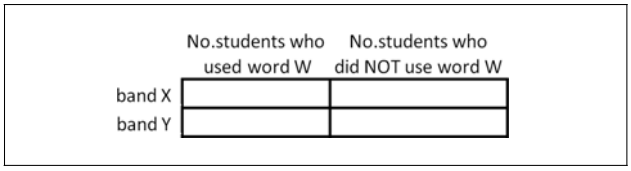

Figure 1: A Contingency Table for two performance bands

3 RESULTS AND DISCUSSION

Table 1 summarizes the performance of the students on all twelve questions, both overall performance, and performance broken down according to how many questions students answered correctly. For example, the third column shows the performance of students on question 25. The row under "Q25" contains the word "swap", a mnemonic for what the code in Q25 does (i.e. it swaps the values between two variables). Below the mnemonic is a percentage, "54%", indicating the percentage of the total student cohort who answered this question correctly. Below that percentage is "10th", indicating this question ranked as the 10th easiest (i.e. 3rd hardest) for the entire cohort. Below that rank is a percentage, "30%", indicating that this "swap" question was answered correctly by 30% of the 27 students who scored 1 out of 12 (i.e. 8 students).

As the top header row in Table 1 shows, the first six EiPE questions involve non-iterative code, while the second six involve iterative code. The row that ranks the ease of each question shows that

non-iterative questions are not necessarily ranked easier than iterative questions. Students who scored from 1 to 3 out of 12 tended to perform much better on non-iterative questions than on iterative questions, but as scores increase the relative ease of iterative versus non-iterative questions is more mixed.

Table 2 contains data similar to Table 1, but Table 2 only contains the data for the three performance bands studied below (i.e. the students who scored 4-7, 9, or 12 out of 12). There is not the obvious performance difference between iterative and non-iterative questions in Table 2 that there is in Table 1. For band 4-7, the average performance on all six non-iterative questions is 54% and for the iterative questions it is 40%. For band 9, the average performance on the non-iterative questions is 68%, but the overall performance on the iterative questions is actually better, at 82%.

On the basis of Tables 1 and 2, we can already answer our second research question: some EiPE questions are harder than others. However, it is not clear what makes some EiPE questions harder than others. Students who performed poorly on all the EiPE questions tended to find the iterative code questions harder to explain than non-iterative code, but that tendency reversed for students who performed well on the EiPE questions.

3.1 Length of Correct, Relational Answers

One of the claims from LIWC research is that experts provide longer answers than novices [7]. If that is also the case with the data used in this study (i.e. that students who scored a perfect 12 provided longer answers), then any difference in word frequencies between performance bands might be attributable to better-performing students providing longer answers. However, Table 3 shows that for each EiPE question, there is little difference in the average number of words used in correct, relational answers by students in different performance bands. With one exception, ANOVA tests revealed no statistically significant differences in the length of student answers between the three performance bands. The exception was Q26, between the bands 4-7 and 12. That exception will be discussed in the subsection below that describes Q26.

3.2 Q25: Swapping Two Variables ("swap")

The code for Q25 is shown in Figure 2. A correct answer is "it swaps the values in variables b and c". For an answer to be correct, a student had to be clear about which two variables have their values swapped. It was not essential for a student to mention that variable "a" was used to hold a temporary value.

This EiPE question has been used in several earlier published studies [2, 14, 17, 18, 21]. Corney et al. [2] gave swapping a special status, describing it as being the "Hello World of Relational Reasoning". This question would probably be easier if the variable name "a" were replaced with something more descriptive, like "temp". However, using an obscure variable name for the temporary variable is consistent with earlier studies [14, 18, 21]. (But the reader will see that the use of the variable name "temp" in the next question, Q26, did not make that question easy.) It also should be noted that this question may have been easier for the students in this study than this question would be for students at other institutions, since swapping using exactly this code featured prominently in the first two weeks of their lectures.

Table 4 shows the frequency with which various words were used by each of the three performance bands. In fact, as explained below, Table 4 shows that there is no statistically significant difference in the word usage of the three performance bands, perhaps because there is little scope for variation in correct answers for this particular question. However, Table 4 will now be discussed in detail so that the reader will be able to understand the subsequent tables for the other EiPE questions.

```
a = b;
b = c;
c = a;
```

Figure 2: The Code Used in Question 25 ("swap")

The first column of Table 4, headed "Word", lists some of the more common words used in correct student answers. All words listed in the first column are by convention shown in capitals, irrespective of how students used capitalisation. The second column, headed "All", shows the percentage of correct answers that contained that word for all the students in the three performance bands combined. For example, the variable names 'B' and 'C' occur in 99% of all correct answers. (These two variable names do not occur in all correct answers because an answer like "it swaps two of the variables using 'a' as a temporary variable" would be marked as correct: since that answer implies which variables have their values swapped.) The rows in the table are ordered by the percentage in the "All" column and thus the words listed in the first few rows of the table – 'SWAP', 'VALUES', 'B', 'C' – approximate a common correct answer. (The words listed in the first few rows of most of the subsequent word frequency tables similarly approximate a common correct answer.)

In Table 4, the three columns headed "4-7", "9" and "12" show, in each row, the percentage of correct answers that included the word in the first column, for students in the three performance bands. For example, the variable name 'A' was mentioned by 33% of students who scored 4-7, 20% of students who scored 9, and 56% of students who scored 12. Separating these three columns containing those percentages are two columns, both headed "Sig". These "Sig" columns show the degree of statistical significance in the difference between the two percentages separated by a "Sig" column. In Table 4, all rows for both "Sig" columns contain "ns", indicating that there are no statistically significant differences in the percentages shown in this table, using the traditional criterion of $p < 0.05$.

The third column of Table 4, headed "p", shows in each row the probability from a chi-square test of whether there is a significant statistical difference in the percentages across all three performance bands. In this table, none of those "p" values are less than 0.05.

3.3 Q26: Shifting Array Values Right ("shiftR")

The code for Q26 is shown in Figure 3. A minimal correct answer is "it shifts one position". A student did not have to specify that the shift was to the right. In fact, an answer was correct even if a student specified a shift left. Also, it was not essential for a student to mention that the rightmost value (or leftmost) wrapped around. (The statistically significant difference in answer length between

Table 1: Performance of the Students on the Twelve EiPE Questions, Q25 to Q36

		Non-Iterative						Iterative					
Student Score	**Number of Students**	**Q25 swap** 54% 10th	**Q26 shiftR** 48% 12th	**Q27 rev** 74% 2nd	**Q28 large3** 78% 1st	**Q29 sort3** 63% 6th	**Q30 mid3** 73% 3rd	**Q31 add1** 62% 7th	**Q32 sum** 67% 5th	**Q33 sorted** 59% 8th	**Q34 ftnd** 68% 4th	**Q35 possum** 52% 11th	**Q36 same** 54% 9th
1	27	30%	11%	26%	11%	19%	0%	0%	0%	0%	0%	0%	4%
2	15	33%	7%	73%	47%	13%	20%	7%	0%	0%	0%	0%	0%
3	18	56%	28%	61%	56%	33%	17%	0%	11%	6%	11%	6%	17%
4	12	33%	8%	67%	83%	75%	50%	33%	8%	17%	0%	0%	25%
5	18	50%	17%	67%	67%	72%	67%	56%	28%	22%	28%	17%	11%
6	20	20%	20%	55%	70%	55%	75%	40%	70%	45%	80%	30%	40%
7	19	37%	37%	63%	74%	42%	95%	42%	84%	32%	74%	58%	63%
8	21	48%	29%	76%	90%	62%	86%	86%	67%	71%	90%	48%	48%
9	39	38%	49%	72%	97%	67%	87%	77%	97%	90%	95%	69%	62%
10	35	57%	80%	86%	89%	74%	91%	94%	94%	86%	97%	83%	69%
11	43	86%	77%	95%	100%	84%	100%	95%	100%	93%	100%	79%	91%
12	36	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 2: Performance of the Students on the Twelve EiPE Questions, Q25 to Q36, for the three performance bands

		Non-Iterative						Iterative					
Student Score	**Number of Students**	**Q25 swap** 52% 11th	**Q26 shiftR** 49% 12th	**Q27 rev** 74% 5th	**Q28 large3** 86% 1st	**Q29 sort3** 72% 6th	**Q30 mid3** 84% 2nd	**Q31 add1** 67% 7th	**Q32 sum** 76% 3rd	**Q33 sorted** 64% 8th	**Q34 ftnd** 75% 4th	**Q35 possum** 58% 10th	**Q36 same** 59% 9th
4-7	69	35%	22%	62%	72%	59%	74%	43%	52%	30%	51%	29%	36%
9	39	38%	49%	72%	97%	67%	87%	77%	97%	90%	95%	69%	62%
12	36	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 3: Average Number of Words in Correct Answers by each Performance band for each EiPE Question

Question	**No. of students**			**Avg. no. of words**		
	4-7	**9**	**12**	**4-7**	**9**	**12**
Q25 swap	24	15	36	11.5	9.8	12.3
Q26 shiftR	15	19	36	19.5	21.3	21.8
Q27 rev	51	34	36	10.5	8.4	12.5
Q28 large3	50	38	36	12.1	10.9	14.7
Q29 sort3	41	26	36	15.1	13.1	20.0
Q30 mid3	51	34	36	16.5	13.5	20.9
Q31 add1	30	30	36	15.3	11.7	11.4
Q32 sum	49	38	36	11.7	11.6	14.6
Q33 sorted	21	35	36	19	18.4	23.9
Q34 find	35	37	36	24.1	22.3	27.8
Q35 possum	20	27	36	15.9	14	16.5
Q36 same	25	24	36	20.3	14.6	20.8

band 12 and band 4-7 is probably because higher band students were more likely to mention these details.) Even with this generous grading, Tables 1 and 2 indicate this was one of the more di@cult questions – despite this code being prominent in the first three weeks of lectures and being part of a week 4 lab test.

Table 4: Word Frequencies for Q25 ("swap")

Word	**All**	**p**	**4-7**	**Sig**	**9**	**Sig**	**12**
'B'	99%	0.341	96%	ns	100%	ns	100%
'C'	99%	0.341	96%	ns	100%	ns	100%
VALUES	88%	0.373	88%	ns	100%	ns	78%
SWAP	88%	0.818	96%	ns	93%	ns	75%
'A'	36%	0.427	33%	ns	20%	ns	56%
TEMP	24%	0.301	25%	ns	7%	ns	39%

Assume that the "x" is an array of three integers.

```
temp = x[2];
x[2] = x[1];
x[1] = x[0];
x[0] = temp;
```

Figure 3: The Code Used in Question 26 ("shiftR")

Table 5 shows that few of the words commonly used exhibit a statistically significant difference between the three performance bands. The double asterisks in the row for the word 'LAST' indicates a statistically significant difference, with $p < 0.01$, between band

Table 5: Word Frequencies for Q26 ("shiftR")

Word	All	p	4-7	Sig	9	Sig	12
RIGHT	81%	0.573	87%	ns	74%	ns	83%
ONE	58%	0.955	60%	ns	58%	ns	56%
SHIFT	57%	0.649	53%	ns	53%	ns	64%
POSITION	49%	0.476	47%	ns	42%	ns	58%
LAST	43%	0.017 *	20%	**	68%	ns	42%
IN	42%	0.044 *	20%	ns	47%	ns	58%
FIRST	34%	0.348	20%	ns	42%	ns	39%
RIGHTMOST	25%	0.203	33%	ns	11%	ns	31%
LEFTMOST	19%	f 0.111	33%	ns	5%	ns	19%
LEFT	13%	0.937	13%	ns	11%	ns	14%

4-7 and band 9. In this case, the difference is probably because students in band 4-7 were less likely to describe how an element on the end of the array wraps around. The asterisk in the row for the word 'IN', under the column headed "p", indicates that there is a statistically significant difference ($p < 0.05$) across all three performance bands. In fact, although it is not shown in the table, the difference is between band 4-7 and band 12 ($p = 0.012$).

3.4 Q27: Reversing Array Values ("rev')

The code for Q27 is shown in Figure 4. A correct answer is "reverse the array order". It was not acceptable for a student to say that the code swapped elements 0 and 4, and elements 1 and 3. Despite this grading requirement, Tables 1 and 2 show that students found this third question much easier than the first two questions.

Table 6 shows that students who scored a perfect 12 were more likely to mention the array name, 'X'. This explicit reference to a variable by the high achieving students is something seen in several more questions, but the authors have no explanation for it.

Assume that the "x" is an array of five integers.

```
temp = x[4];
x[4] = x[0];
x[0] = temp;

temp = x[3];
x[3] = x[1];
x[1] = temp;
```

Figure 4: The Code Used in Question 27 ("rev")

Table 6: Word Frequencies for Q27 ("rev")

Word	All	p	4-7	Sig	9	Sig	12
REVERSES	95%	0.324	98%	ns	89%	ns	97%
ARRAY	85%	0.078	77%	ns	93%	ns	86%
ORDER	68%	0.772	60%	ns	68%	ns	75%
'X'	32%	0.005 **	14%	ns	18%	*	64%

3.5 Q28: Largest of Three Values ("large3")

The code for Q28 is shown in Figure 5. A correct answer is "it prints the largest value". This question has been used in two earlier published studies [13, 21].

Table 7 shows that, as was the case in the previous question, the students who scored a perfect 12 were more likely to mention the variable names (e.g. "it prints the largest value in variables a, b and c"). The statistics for 'A', 'B' and 'C' are identical, so the three variable names are shown on a single row of Table 7.

```
if ( a < b)
{
   if ( b < c) System.out.println
            (c);
   else
        System.out.println (b);
}
 else
{
   if ( a < c) System.out.println
            (c);
   else
        System.out.println (a);
}
```

Figure 5: The Code Used in Q28 ("large3")

Table 7: Word Frequencies for Q28 ("large3")

Word	All	p	4-7	Sig	9	Sig	12
PRINT	91%	0.541	88%	ns	95%	ns	92%
LARGEST	68%	0.4	60%	ns	71%	ns	72%
'A' 'B' 'C'	57%	0.002 **	44%	ns	47%	**	81%
VARIABLE	51%	0.057	48%	ns	39%	*	67%
HIGHEST	16%	0.994	16%	ns	16%	ns	17%
BIGGEST	11%	0.704	14%	ns	11%	ns	8%

3.6 Q29: Sort Three Values ("sort3")

The code for Q29 is shown in Figure 6. A correct answer is "it sorts the values in the variables". Students did not need to specify that the code sorted the values into descending order. In fact, students were graded as having a correct answer even if they asserted incorrectly that the values were sorted into ascending order. This question has been used in several earlier published studies [2, 17–19].

Table 8 shows that, as was the case for the previous two questions, the students who scored a perfect 12 were more likely to mention the variable names (e.g. "it sorts the values in y1, y2 and y3").

3.7 Q30: Middle of Three Values ("mid3")

The code for Q30 is shown in Figure 7. A correct answer is "it prints the middle value". Many students, in all three bands, gave correct answers that did not mention "middle", such as "It prints the value that is not the minimum or the maximum". This question was used in one earlier study [18].

Table 9 shows that, as was the case for the previous three questions, the students who scored a perfect 12 were more likely to mention the variable names, 'FIRST', 'SECOND' and 'THIRD'. The band 12 students were also more likely to use the word "middle", perhaps indicating a greater willingness to use a word not explicitly in the code, but instead an abstraction beyond the code itself (as may also be the case in Q34 and Q35).

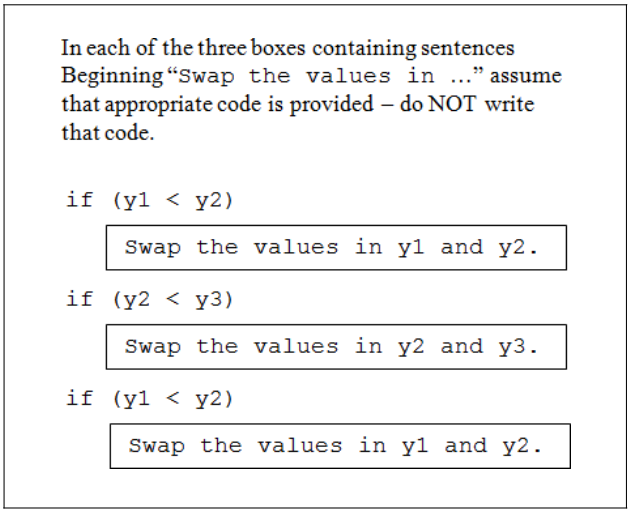

Figure 6: The Code Used in Q29 ("sort3")

Table 8: Word Frequencies for Q29 ("sort3")

Word	All	p	4-7	Sig	9	Sig	12
VALUE	67%	0.427	71%	ns	58%	ns	72%
ORDER	65%	0.621	59%	ns	69%	ns	67%
'Y1'	54%	0.009 **	44%	ns	42%	**	75%
TO	54%	0.01 **	71%	ns	54%	ns	36%
'Y3'	51%	0.003 **	44%	ns	35%	**	75%
'Y2'	40%	0.019 *	39%	ns	23%	**	58%
SORT	40%	0.301	34%	ns	35%	ns	50%
FROM	39%	0.031 *	51%	ns	42%	ns	22%

3.8 Q31: Add 1 to Array Values ("add1")

The code for Q31 is shown in Figure 8. A correct answer is "it adds 1 to every element". This question has not been used in any earlier empirical study of EiPE questions, but it was asserted by Gluga et al. [5] that this piece of code is an especially easy piece of iterative code for a novice to explain. However, Tables 1 and 2 show that students did not find this question to be easy in comparison to other questions involving iterative code. A common incorrect answer was that the code shifted the elements by one position. Such an incorrect answer may stem from the well-known problem novices have of distinguishing between a position in an array and the contents of that position [4].

Table 10 shows that, as was the case for the previous four questions, the students who scored a perfect 12 were more likely to mention the variable name, 'X'.

3.9 Q32: Sum Array Values ("sum")

The code for Q32 is shown in Figure 9. A correct answer is "It prints the sum of all the values in the array". This question was used in three previous studies [3, 11, 15].

Table 11 shows the frequency with which various words were used by students in the three performance bands. Words such as 'SUM' and 'PRINT' do not show high percentages because many students used synonyms, such as 'TOTAL' and 'DISPLAY'.

The variable *maximum* contains the maximum value stored in variables *first*, *second* and *third*.

The variable *minimum* contains the minimum value stored in variables *first*, *second* and *third*.

```
if (first == minimum)
{
   if (second == maximum) System.out.println(third);
   else
        System.out.println(second);
}
 else
{
   if (second == minimum)
   {
      if (first == maximum) System.out.println(third);
      else
           System.out.println(first);
   }
   else
   {
      // if we get to here,
      // then minimum == third
      if (first == maximum) System.out.println(second);
      else
           System.out.println(first);
   }
}
```

Figure 7: The Code Used in Q30 ("mid3")

Table 9: Word Frequencies for 30 ("mid3")

Word	All	p	4-7	Sig	9	Sig	12
VALUE	77%	0.029 *	82%	*	62%	*	86%
MIDDLE	64%	0.053	63%	ns	50%	*	78%
MINIMUM	35%	0.245	25%	ns	41%	ns	39%
MAXIMUM	32%	0.487	25%	ns	35%	ns	36%
'THIRD'	29%	0.003 **	24%	ns	15%	**	50%
'FIRST'	29%	0.003 **	24%	ns	15%	**	50%
'SECOND'	29%	0.003 **	24%	ns	15%	**	50%

An interesting feature of the word frequencies is that, while there is no statistically significant difference between the bands in their use of the name of the array, 'X', the students in band 4-7 are much more likely to mention the auxiliary variable, 'Z', which may be an indication that students in band 4-7 are more focussed on the process of calculating the result, whereas students in the higher bands are more focussed on the final result.

```
for (int i=0 ; i<x.length ; ++i ) x[i] =
    x[i] + 1;
```

Figure 8: The Code Used in Question 31 ("add1")

Table 10: Word Frequencies for Q31 ("add1")

Word	All	p	4-7	Sig	9	Sig	12
1	74%	0.207	83%	ns	63%	ns	75%
'X'	47%	0.015 *	40%	ns	33%	**	67%
ADDED	41%	0.26	50%	ns	43%	ns	31%
INCREMENT	32%	0.522	40%	ns	27%	ns	31%
ELEMENT	21%	0.027 *	10%	ns	17%	ns	36%
ONE	20%	0.225	13%	ns	30%	ns	17%
INCREASE	16%	0.202	7%	ns	20%	ns	22%

```
int z = 0;
for (int i=0 ; i<x.length ; ++i ) z = z +
    x[i];
System.out.println(z);
```

Figure 9: The Code Used in Q32 ("sum")

Table 11: Word Frequencies for Q32 ("sum")

Word	All	p	4-7	Sig	9	Sig	12
ARRAY	90%	0.637	92%	ns	92%	ns	86%
SUM	79%	0.021 *	64%	*	84%	ns	89%
PRINT	74%	0.079	72%	ns	63%	*	86%
IN	63%	0.066	58%	ns	53%	*	78%
ALL	61%	0.237	56%	ns	55%	ns	72%
'X'	55%	0.091	47%	ns	47%	ns	69%
'Z'	26%	0.005 **	44%	**	13%	ns	19%

3.10 Q33: Check if an Array is Sorted ("sorted")

The code for Q33 is shown in Figure 10. A correct answer is "It checks if the array is sorted". The student did not have to specify that the code checked for ascending order. In fact, a student's answer was marked as correct even if the student specified descending order. Code like this has been used in many studies with EiPE questions [3, 9, 10, 14, 22]. The results in this paper are the first time that student performance on this EiPE question has been compared to several other EiPE questions. Tables 1 and 2 show that this EiPE question ranks as the 8th easiest of the 12 EiPE questions.

Table 12 shows the frequency with which various words were used by students in the three performance bands. The band 12 students are more likely to use 'RETURN', '1' and '0'.

```
public static int validate(int iNumbers[])
{
    int bValid = 1;

    for (int i=0; i<iNumbers.length-1; i++)
    {
        if (iNumbers[i] > iNumbers[i+1])
        {
            bValid = 0;
        }
    }

    return bValid;
}
```

Figure 10: The Code Used in Q33 ("sorted")

Table 12: Word Frequencies for Q33 ("sorted")

Word	All	p	4-7	Sig	9	Sig	12
IF	87%	f 0.067	81%	ns	83%	ns	97%
ARRAY	85%	0.432	81%	ns	83%	ns	92%
RETURN	67%	0.047 *	57%	ns	60%	*	83%
1	66%	0.012 *	52%	ns	60%	*	86%
0	66%	0.009 **	57%	ns	54%	**	86%
ASCENDING	54%	0.409	48%	ns	51%	ns	64%
'INUMBER'	40%	0.101	29%	ns	37%	ns	56%
SORT	23%	0.127	14%	ns	20%	ns	36%
DESCENDING	18%	0.852	14%	ns	20%	ns	19%

3.11 Q34: Find a Value in an Array ("ftnd")

The code for Q34 is shown in Figure 11. A correct answer is "It finds a value in an array". This code was used in two previous studies [3, 13] but perhaps not with the relaxed grading criteria used in this study. To have an answer graded as correct in this study, a student only needed to articulate that the code was performing some kind of search of the array. The student did not need to specify three important aspects of the code:

(1) if the search value is found, then the method returns the position of that value.
(2) the method returns -1 if the search value is not found.
(3) if the search value occurs more than once in the array, it is the final position of the search value that is returned.

The relaxed grading criteria led to this question being the fourth easiest question, as shown in Tables 1 and 2. Furthermore, while

this specific code was new to the students, the students had been taught linear search in lectures and had completed a lab test on an implementation of linear search during the semester. While the code in this EiPE question was superficially quite different from what was studied and tested during semester (e.g. the code studied and tested during semester used a while loop), the code studied during semester did handle the first two of the three aspects itemized above.

```
public static int q34(int data[], int x )
{
   int z = -1;
   for (int i=0; i < data.length; i++ )
   {
      if ( data[i] == x ) z = i;
   }
   return z;
}
```

Figure 11: The Code Used in Question 34 ("ftnd")

Table 13: Word Frequencies for Q34 ("ftnd")

Word	All	p	4-7	Sig	9	Sig	12
ARRAY	85%	0.368	80%	ns	84%	ns	92%
'X'	83%	0.415	86%	ns	76%	ns	86%
RETURN	80%	0.009 **	66%	ns	81%	ns	94%
-1	70%	0.009 **	51%	*	76%	ns	83%
POSITION	70%	0.012 *	69%	ns	54%	**	86%
FOUND	38%	<0.001 ****	11%	**	41%	ns	61%
EQUAL	24%	0.084	37%	*	16%	ns	19%
FOR	21%	0.073	9%	*	30%	ns	25%
ELEMENT	20%	0.099	26%	*	8%	ns	25%
LAST	14%	0.051	6%	ns	11%	ns	25%

A student might provide a correct answer for this question by only understanding some of the code, which acts as beacons [1]. The two beacons for this question are probably the "for" loop header (which in the experience of these students always run across an entire array) and the complete "if" statement. Even if the rest of the code is simply ignored by a student, the student might provide a correct answer just from those two beacons. Traynor, Bergin, and Gibson [20] noted something similar about the linear search, but as a code-writing exercise. In an interview they conducted with a student, the student described his approach to answering coding questions in an exam:

> . . . you usually get the marks by making the answer look correct. Like, if its a searching problem, you put down a loop and you have an array and an if statement. That usually gets you the marks . . . not all of them, but definitely a pass.

Table 13 shows the frequency with which various words were used by students in the three performance bands. With respect to the three aspects of the code enumerated above: (1) the band 12 students were much more likely to use the word 'POSITION'; (2) the band 4-7 students were much less likely to use '-1'; and (3) the band 12 students were more likely to use the word 'LAST' than the band 4-7 students ($p = 0.025$, which is not listed in Table 13).

An aspect of Table 13 is that band 4-7 students were less likely to use the word 'FOUND' but more likely to use the word 'EQUAL'. An inspection of complete student answers may explain this interesting issue – having deduced what the code does, higher band students are more likely to use forms of expression that abstract from the code itself, to describe the code in terms of what the code does, whereas lower band students tend to use words that are a closer reading of the code itself (as may well be the case in Q30, with the word "middle", and also in Q35). Specifically, having deduced that the code searches the array, higher band students are more inclined to talk about the value being "found", whereas lower band students are more likely to talk about the value in the array being "equal" to the search value. This may be an important clue as to why some students perform more consistently than other students on code explaining, and perhaps also on code writing – students who more actively abstract from the code (however modestly) may more quickly learn general lessons from specific code examples.

3.12 Q35: Sum Positive Values ("possum")

The code for Q35 is shown in Figure 12. A correct answer is "It sums all the positive values in an array". Students had to specify that only positive values are summed. This question was one of the most di@cult. This question was used in one previous study [13], in which those authors described a surprising feature of their students' incorrect answers:

> . . . a common mistake was to respond that the code summed all the elements of the array. In making that mistake, students ignored the if statement within the loop – to do so is an egregious error.

Surprisingly, many students in our study made that same "egregious" error. That surprising result for Q35, in both our study and the previous study, plus our observations about beacons [1] in the previous question, have led us to propose an explanation for this surprisingly common error. In the case of this EiPE question, a student may understand only some of the code. The two critical lines, or beacons, for this question, are probably the "for" loop header (as was the case in the previous question) and the line "num = num + numbers[i]". If the rest of the code is simply ignored by a student, then that student is likely to guess – incorrectly, for this question – that the code sums the values of all elements in the array, not just the positive values. Thus an EiPE question like this question may expose students who are not reading all the code, and who are guessing based on the parts of the code they do understand.

Table 14 shows the same issue that we noted for the words 'MIDDLE' in Q30 and 'FOUND' in Q34 – higher band students are more likely to use words that abstract from the code, whereas lower band students tend to use words that are a closer reading of the code. Specifically, the band 12 students are more inclined to talk about values in the array that are "positive". The band 4-7 students have

a variety of forms for expressing the same thing, and that variety prevents any statistically significant forms from being manifested in Table 14. One of those non-statistically significant forms is in three words – 'GREATER', 'THAN', '0'. The p-values for the differences in the usage of those three words between band 4-7 students and band 12 students are 0.096, 0.033* and 0.096 respectively.

```
public static int q35( int[] numbers )
{
    int num = 0;

    for(int i=0; i < numbers.length; i++ )
    {
       if ( numbers[i] > 0 )
            num = num + numbers[i];
    }

    return num;
}
```

Figure 12: The Code Used in Q35 ("possum")

Table 14: Word Frequencies for Q35 ("possum")

Word	All	p	4-7	Sig	9	Sig	12
ARRAY	88%	f 0.009 **	80%	ns	85%	f *	100%
ALL	72%	<0.001 ***	45%	**	89%	ns	83%
SUM	71%	0.006 **	50%	ns	74%	ns	89%
IN	70%	0.047 *	60%	ns	63%	*	86%
POSITIVE	55%	0.008 **	40%	ns	48%	*	78%
RETURN	66%	0.204	55%	ns	67%	ns	78%
THAN	38%	0.084	50%	ns	41%	ns	22%
0	31%	0.221	40%	ns	33%	ns	19%
GREATER	28%	0.247	40%	ns	26%	ns	19%
ADDED	18%	0.101	30%	ns	15%	ns	8%

3.13 Q36: Count Same Values ("same")

The code for Q36 is shown in Figure 13. A correct answer is "It prints the number of times that the same number occurs in both arrays". This question was the ninth easiest question, as shown in Tables 1 and 2. An earlier exam question (Q18) used the same code for a tracing exercise, where students had to identify the final value in the variable "count", when given specific initializations for the two arrays, "x1" and "x2".

Table 15 shows the frequency with which various words were used by students in the three performance bands. The popularity of the word 'OUT' for band 4-7 students is because they are more likely to write "print out" instead of just "print". On one hand, writing "print out" may not be a fundamental difference in word usage, but on the other hand, perhaps the use of "out" is due to its origins in "System.out.println". Band 4-7 students are much more likely to make explicit reference to the variable "count". (Note that the variable name 'COUNT' appears in the bottom row of the table, and the verb "COUNT" appears near the middle of the table. Recall that variable names are indicated using single quotes.) That difference may indicate that lower band students are more focussed on the actual calculation, while higher band students are more focussed on the result of the calculation.

Hint: **This code is the same code from multiple choice question Q18**, except the declaration and initialization of the arrays "x1" and "x2" are not given here. Go back and look at your answer for Q18. It might help you answer this question.

Note: Unlike Q18, the arrays "x1" and "x2" used here in Q36 may contain any integer values, and be of any length, but with the following restrictions:

1. The elements in each array are sorted in ascending order (i.e. from smallest to largest).
2. No number occurs more than once in the same array.

```
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
   {
      // if we get here, then
      //    x1[i1] > x2[i2]

       --i1;
   }

} // while

System.out.println("count = " + count);
```

Figure 13: The code Used in Q36 ("same")

Table 15: Word Frequencies for Q36 ("same")

Word	All	p	4-7	Sig	9	Sig	12
NUMBER	50%	0.049 *	36%	ns	46%	ns	67%
TIMES	44%	0.866	48%	ns	42%	ns	42%
COUNT	43%	0.607	36%	ns	50%	ns	44%
PRINT	38%	0.802	40%	ns	33%	ns	42%
OUT	16%	f 0.040 *	32%	ns	8%	ns	8%
'COUNT'	14%	f 0.005 **	32%	f *	4%	ns	6%

4 CONCLUSION

This study is the first to compare so many Explain in Plain English questions on the same student cohort. The results in this paper provide a foundation upon which future workers may characterize the di@culty of their own questions, by giving to their students a judicious mix of their new questions and the questions in this paper.

That students use many of the same words in correct answers, with similar frequencies, is not surprising. There are only a limited number of ways of answering many of these questions correctly. In this respect, the results in this paper are different from what has been reported in LIWC studies [6, 7]. This difference is possibly due to the LIWC studies' use of extended responses and reflections, whereas the student responses to Explain in Plain English questions are often just a single sentence.

However, despite the brevity of student answers in this study, there are some differences in how students use words, for some of the questions. Students who score more highly on all twelve questions are more meticulous in their word usage. They are more likely to use words that abstract beyond the code. They are more likely to provide a more comprehensive explanation of all possible behaviours of the code. Also, and inexplicably, they are more likely to make explicit reference to variables, especially in non-iterative code.

The results in this paper are correlations. We cannot infer causality. Did the high scoring students in this study acquire their ways of expressing themselves as part of learning to program, or did they come into the subject already having the inclination to provide these types of answers? We do not know. Either way, the higher scoring students have ways of expressing themselves that lower-scoring students could usefully acquire.

Even if teachers do not use Explain in Plain English questions, the results in this paper might outline the words that teachers should listen for when students describe their code. Through those words, a teacher might diagnose the developmental stage of a student.

REFERENCES

[1] Ruven E. Brooks. 1983. Towards a Theory of the Comprehension of Computer Programs. *International Journal of Man-Machine Studies* 18, 6 (1983), 543–554. http://dblp.uni-trier.de/db/journals/ijmms/ijmms18.html#Brooks83

[2] Malcolm Corney, Raymond Lister, and Donna Teague. 2011. Early Relational Reasoning and the Novice Programmer: Swapping As the "Hello World" of Relational Reasoning. In *Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114 (ACE '11)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 95–104. http://dl.acm.org/citation.cfm?id=2459936.2459948

[3] Malcolm Corney, Donna Teague, Alireza Ahadi, and Raymond Lister. 2012. Some Empirical Results for neo-Piagetian Reasoning in Novice Programmers and the Relationship to Code Explanation Questions. In *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123 (ACE '12)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 77–86. http://dl.acm.org/citation.cfm?id=2483716.2483726

[4] B. DuBoulay. 1989. Some Di@culties of Learning to Program. In *Studying the Novice Programmer*, E. Soloway and J. C. Spohrer (Eds.). Number 14. Lawrence Erlbaum Associates, 283–299. http://www.sussex.ac.uk/Users/bend/papers/diffsofprogramming.pdf

[5] Richard Gluga, Judy Kay, Raymond Lister, and Donna Teague. 2012. On the Reliability of Classifying Programming Tasks Using a Neo-piagetian Theory of Cognitive Development. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research (ICER '12)*. ACM, New York, NY, USA, 31–38. https://doi.org/10.1145/2361276.2361284

[6] Camilla Gobbo and Michelene Chi. 1986. How knowledge is structured and used by expert and novice children. *Cognitive Development* 1, 3 (July 1986), 221–237. https://doi.org/10.1016/S0885-2014(86)80002-8

[7] Kyungil Kim, Jinhee Bae, Myung-Woo Nho, and Chang Hwan Lee. 2011. How do experts and novices differ? Relation versus attribute and thinking versus feeling in language use. *Psychology of Aesthetics, Creativity, and the Arts* 5, 4 (July 2011), 379–388. https://doi.org/10.1037/a0024748

[8] Changhwan Lee, Kyungil Kim, Jeongsub Lim, and Yoonhyoung Lee. 2015. Psychological Research using Linguistic Inquiry and Word Count (LIWC) and Korean Linguistic Inquiry and Word Count (KLIWC) Language Analysis Methodologies. *Journal of Cognitive Science* 16, 2 (2015), 132–49. http://cogsci.snu.ac.kr/jcs/index.php/issues/?pageid=4&mod=document&uid=183

[9] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship Between Explaining, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09)*. ACM, New York, NY, USA, 161–165. https://doi.org/10.1145/1562877.1562930

[10] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L. Whalley, and Christine Prasad. 2006. Not Seeing the Forest for the Trees: Novice Programmers and the SOLO Taxonomy. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITICSE '06)*. ACM, New York, NY, USA, 118–122. https://doi.org/10.1145/1140124.1140157

[11] 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

[12] David S. Moore, George P. McCabe, and Bruce A. Craig. 2017. *Introduction to the Practice of Statistics* (9 ed.). w.h. freeman, New York, NY, USA. 10–15 pages.

[13] 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

[14] Judy Sheard, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, and Jacqueline L. Whalley. 2008. Going SOLO to Assess Novice Programmers. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '08)*. ACM, New York, NY, USA, 209–213. https://doi.org/10.1145/1384271.1384328

[15] Simon and Susan Snowdon. 2011. Explaining Program Code: Giving Students the Answer Helps - but Only Just. In *Proceedings of the Seventh International Workshop on Computing Education Research (ICER '11)*. ACM, New York, NY, USA, 93–100. https://doi.org/10.1145/2016911.2016931

[16] Yla R. Tausczik and James W. Pennebaker. 2009. The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. *Journal of Language and Social Psychology* 29, 1 (dec 2009), 24–54. https://doi.org/10.1177/0261927x09351676

[17] Donna Teague, Malcolm Corney, Alireza Ahadi, and Raymond Lister. 2012. Swapping As the "Hello World" of Relational Reasoning: Replications, Reflections and Extensions. In *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123 (ACE '12)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 87–94. http://dl.acm.org/citation.cfm?id=2483716.2483727

[18] Donna Teague, Malcolm Corney, Alireza Ahadi, and Raymond Lister. 2013. A Qualitative Think Aloud Study of the Early Neo-piagetian Stages of Reasoning in Novice Programmers. In *Proceedings of the Fifteenth Australasian Computing Education Conference - Volume 136 (ACE '13)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 87–95. http://dl.acm.org/citation.cfm?id=2667199.2667209

[19] 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

[20] Des Traynor, Susan Bergin, and J. Paul Gibson. 2006. Automated Assessment in CS1. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE '06)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 223–228. http://dl.acm.org/citation.cfm?id=1151869.1151898

[21] 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

[22] Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P. K. Ajith Kumar, and Christine Prasad. 2006. An Australasian Study of Reading and Comprehension Skills in Novice Programmers, Using the Bloom and SOLO Taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE '06)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 243–252. http://dl.acm.org/citation.cfm?id=1151869.1151901