First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts

James Prather, Raymond Pettit, Brett A. Becker, Paul Denny, Dastyni Loksa, Alani Peters, Zachary Albrecht, Krista Masci

Motivating example

A scene from a typical open lab time in Programming I (CS1):

- Student raises their hand.
- Student: I need some help.
- Professor: Ok, what's going on?
- Student: I don't know.
- Professor: Where are you stuck?
- Student: I don't know. But here's my code. Why isn't it working?

This story anecdotally illustrates that **novice programmers lack metacognitive awareness.**

Metacognition

The story anecdotally illustrates that **novice programmers lack metacognitive awareness.**

- **Metacognitive awareness** is the ability to not only understand the problem but also understand where one is in the problem-solving process and the ability to reflect on that state.
- Previous research shows that:
  - Most novice programmers lack metacognitive awareness, but the highest performing students may already have some of these skills (Bergin, Reilly, & Traynor, 2005)
  - Novice programmers tend to struggle through the problem-solving stages, often repeating them or revisiting them in cycles (Loksa et al., 2016)
  - Novice programmers tend to face metacognitive difficulties when first learning to code (Prather et al., 2018)

Metacognition: Theoretical Frameworks

Loksa's Stages (Loksa et al., 2016)
1. Reinterpret the prompt
2. Search for analogous problems
3. Search for solutions
4. Evaluate a potential solution
5. Implement a solution
6. Evaluate implemented solution

Metacognitive Difficulties (Prather et al., 2018)

Metacognitive Difficulty	Explanation
Forming	Forming the wrong conceptual model about the right problem
Dislodging	Dislodging an incorrect conceptual model of the problem may not be solved by re-reading the prompt
Assumption	Forming the correct conceptual model for the wrong problem
Location	Moving too quickly through one or more stages incorrectly leads to a false sense of accomplishment and poor conception of location in the problem-solving process
Achievement	Unwillingness to abandon a wrong solution due to a false sense of being nearly done

Research Question

Can solving a prescribed test case immediately after reading a problem prompt help novice programmers overcome metacognitive difficulties encountered in the early stages of problem-solving?

Methodology

Think-Aloud Study:

- One-on-one with researcher for 1 hour
- 38 students opted-in to study (all requisite IRB requirements were followed)
- Student given Athene problem and asked to solve it while talking aloud
  - Students given warm-up exercise to help ease them into think-aloud
  - All students used the same coding environment on the lab computer
  - All code was in C++
- Students in the **experimental group** were asked to **solve a test case** before starting to code
- Pre- and post-quiz growth mindset questions
- Observational data, post-session interviews, and submitted code used in qualitative analysis

Quiz: More Positive or Negative?

During this quiz, you may not access or view any other materials, either course notes, previous homework submissions or any online material; nor may you attempt to communicate with any person other than the instructor.

Write a program that prompts the user to enter a series of integer values terminating in 0. When the input is done, report whether there were more positive than negative values ("Positive"), more negative than positive values ("Negative"), or an equal number of positive and negative values ("Equal"). Note that we do not want the sum of the numbers, just a comparison of the relative count of positive vs. negative numbers.

Your program should run like the examples shown below:

```
Enter number: 2
Enter number: -35
Enter number: 7
Enter number: 0

Positive
```

In the first example (above): there are two positive numbers (2,7) and one negative number (-35), so there are more positive numbers than negative, as indicated in the output.

```
Enter number: 0

Equal
```

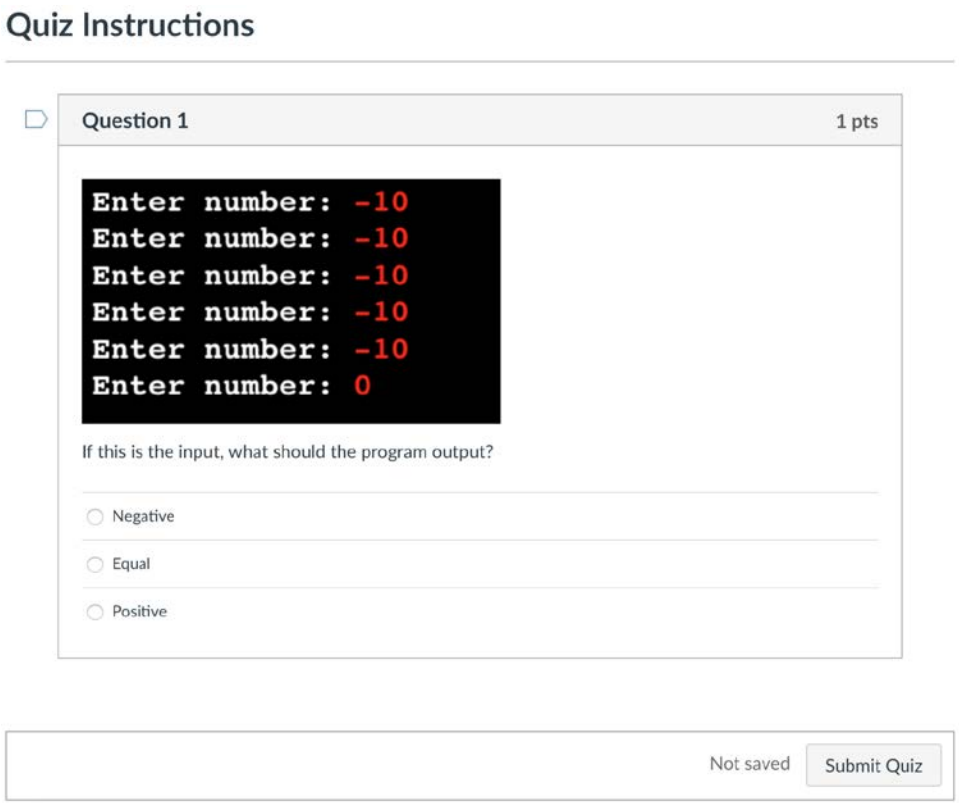
```
Enter number: -3
Enter number: -10
Enter number: -5
Enter number: 0

Negative
```

Note: this program is given as a quiz, part of an ongoing assessment of your progress in the class. It may be graded differently from homework assignments. Athene will provide an estimated score, but your final score will be determined by the instructor upon reviewing your work; the final score may be higher or lower than the automated score.

Solving a Test Case

After reading the problem prompt, participants in the experimental group were asked to solve a random test case.

Here is an example ------->



Post-quiz Interview Questions

1) Please describe how you usually go about solving your Athene homework problems.

If in experimental group:

2) You aren't usually asked to solve a test case before you can start coding. Why do you think you were asked to do that today?
3) What do you think solving a test case before coding did for you?
4) Do you think being asked to solve a test case before coding helped your quiz performance? If so, please describe how.

If they didn't complete the problem (for both experimental AND control groups):

5) Describe where you think you got lost while solving this problem?

Quantitative Results: Submission Data

It appears that the intervention helped more participants complete the programming task compared to those that did not receive the intervention - the experimental group had a higher completion rate, faster time, and fewer attempts required to complete. (although it's difficult to argue for statistical significance given the small number of participants in each group)

	Control	Experimental
Correct completion rate	52.94%	76.19%
Mean time (minutes)	23.82	22.62
Mean code submissions	7.59	4.48

Quantitative Results: Test Case Completion

Experimental Group:

One participant did not correctly solve the test case on the first attempt (P34), solved it on the second attempt, and did not complete program.

Four more participants correctly solved the test case on the first attempt, but did not complete the program.

There is no correlation between number of attempts to solve the test case and program completion.

(Right: "# of Attempts" is the number of times the participant tried to solve a test case before getting it correct and moving on. "Time to Complete" is in minutes.)

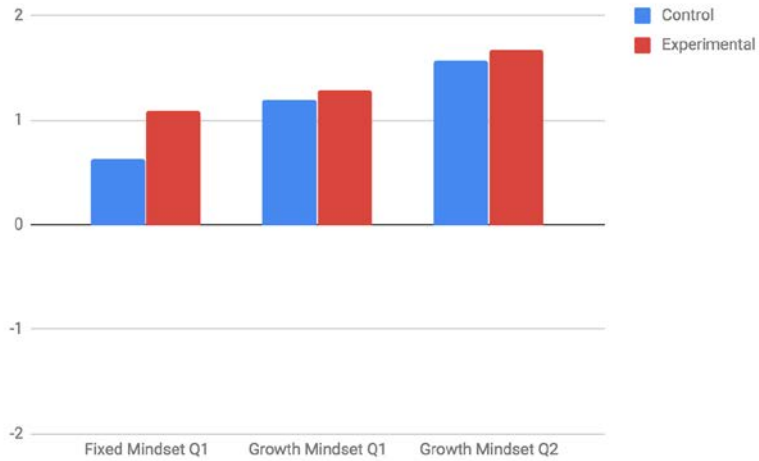
Name	# of Attempts	Time to Complete
P20	1	12
P21	1	30
P22	1	22
P23	1	12
P24	1	28
P25	1	16
P26	1	28
P27	1	9
P28	1	10
P29	1	20
P30	1	13
P31	1	9
P33	1	incomplete
P34	2	incomplete
P35	1	31
P36	1	incomplete
P38	1	28
P39	1	incomplete
P41	1	17
P42	1	incomplete
P43	1	15

Quantitative Results: Growth Mindset

Growth mindset data was inconclusive (p=0.1070):

- Control group average increase: +0.308
- Experimental group average increase: +0.000

Experimental group began with a higher average.



Above: mindset data for both groups before receiving the programming task.

Qualitative Results: Experimental Group (n=21)

Summary: Participants in the experimental group who submitted a correct code solution tended to display and verbalize higher metacognitive skills and behaviors regarding the problem prompt than those in the control group. Participants in this group still faced multiple metacognitive difficulties

Indicative Quotes:

P31 - “[It] helped me understand that when you don’t enter anything it would be equal,”

P43 - “I usually freak out reading the prompt, but doing a test case helped me breathe and know that I know how to do it."

P29 - “[It] made me realize that I didn’t read the problem very well because I needed to go back and read it again before I answered the quiz.”

Qualitative Results: Control Group (n=17)

Summary: Participants in the control group naturally divided into two sub-groups: those that re-read the problem prompt and those that did not. Re-reading the problem prompt appears to be correlated with the *Forming* metacognitive difficulty. Participants in this group also faced multiple metacognitive difficulties

Did not re-read the prompt (10):

- 10 did not re-read the problem prompt
- 9 did not face any metacognitive difficulties
- 4 did not submit a correct code solution:
  - 1 faced the *Forming* metacognitive difficulty
  - 3 faced syntax errors

Re-read the prompt (7):

- 7 re-read the problem prompt at least once
- Participants in this sub-group re-read the problem prompt once (1), twice (1), three times (4), and five times (1).
- **No participants from this sub-group submitted a correct code solution.**

Conclusions

Big Takeaways:

Adding metacognitive scaffolding onto problem prompts forces reflection and appears to improve student problem-solving success.

Re-reading the problem prompt appears to:

1. have a negative correlation with student problem-solving success
2. have a positive correlation with the number and severity of metacognitive difficulties faced

Future Work:

We have already replicated this study at scale (~1,000 participants) and are working on analysis of those results.

This study generated some new questions:

1. How do randomly generated test cases impact studies like this?
2. Can we confirm a correlation between the number of times a student re-reads a problem prompt and the number and severity of metacognitive difficulties faced?

Providing Metacognitive Scaffolding in Your Classroom

So this type of scaffolding works...how can I do it?

- Teach students about the problem-solving stages, how to work through them, and how to identify where they are in that process when they are stuck.
- Use your LMS to build-in reflection on the problem:
  - What is it I'm being asked to do?
  - How do I think I'd accomplish that?
- Use your LMS to test whether they understand the answers to these questions, such as through a test case quiz before being allowed to start coding.
- Learn to recognize the metacognitive difficulties so you can provide appropriate help to your students or build-in automated help into your AAT (if you control it).

Thank you. Questions?

jrp09a@acu.edu