Recursive Thinkers and Doers in Introductory Computer Science

INTRODUCTION

Computer scientists need a precise and complete understanding of recursion. Earlier indicators that recursion merits attention in the first Computer Science classes [2]. We present a simple test question that, when used in a final exam for three courses over two semesters, significantly predicts student performance on the final. We also present an evaluation rubric that reveals misconceptions about recursion and recursive procedures.

STUDY DESIGN

What does it mean for a procedure to be recursive?

We asked students to define recursion on the final exam. We explored the relationship between different answer types and performance on the final (a proxy for overall learning).

Data was collected over two semesters and three sections of the introductory course. The central question was coded by 6 raters in a single session (lasting approximately 3 hours). Each rater coded all 120 responses. Each rater’s coding sheet was in a unique random order. Raters came from a variety of backgrounds.

INTERPRETATION

We characterize encountered responses to the question as form-oriented when it focused on the static structure of the program. In contrast, process-oriented responses focus on the behavior of the program while executing. We find that students who define recursion in a form-oriented fashion (i.e., the recursive thinkers) tended to perform in the A range on the exam. Students who focused more on the process of recursion (i.e., recursive doers) clustered in the B-/C+ range. Students who did not associate recursion with self-reference were in the D range.

REFERENCES


Acknowledgements: The authors thank Claire Alvis, William Byrd, Jennie Lipson, and John-Paul Verkamp for assisting us with the coding process. We thank the Open Systems Laboratory for the time and resources that enabled this research.