We thank the reviewers for their valuable suggestions. Please find our answers for each reviewer (**R**) below.

[**R1, R2: Additional technical details in the paper and appendices**] As suggested by Reviewer 1, we will provide a detailed example of the combination of MCTS and symbolic execution, and add details on Delta debugging. As per Reviewer 2, we will move the details of $\mathcal{F}_{\text{score}}$ and $\mathcal{F}_{\text{qual}}$ to the main paper. Especially, if the paper is accepted, the extra page in the final submission will allow us to incorporate these additional details in the main paper.

[**R1, R2: Pre-specification of domain knowledge**] To answer Reviewer 1's question, the domain knowledge is mostly universal to both HOC and Karel tasks—a few differences arise primarily because Karel's DSL is slightly different (also, see $\mathcal{F}_{\text{qual}}^{\text{HOC}}$ and $\mathcal{F}_{\text{qual}}^{\text{Karel}}$ in L570 and L576 respectively). To answer Reviewer 2, the manual effort in designing the Sketch constraints and cost function was actually minimal, compared to the effort required to manually generate a large pool of new tasks. As suggested, it is an interesting direction to automatically learn the constraints and cost function from a corpus. To apply our methodology to other programming environments (e.g., Python problems), one should first establish the specification of tasks/problems: it would be easier to extend our methodology with tasks specified as I/O pairs.

[**R1, R4: Lack of learning aspects in the solution**] We thank the reviewers for appreciating the importance of the problem and its relevance to the NeurIPS community. Regarding concerns on the lack of learning aspects in the solution, we believe that the richness of the problem setting and obtained results will spur interesting follow-up works requiring more complex learning-based solutions. One exciting direction, as hinted by Reviewer 1, is to learn a policy to guide the MCTS procedure (instead of running vanilla MCTS). Another important direction, as suggested by reviewers, is to automatically learn the constraints and cost function from a human-generated pool of problems.

Beyond the application domain of programming education, our methodology can be used for generating large-scale training datasets consisting of tasks and solution codes with desirable characteristics—this can be potentially useful for researchers in the NeurIPS community, for example, to improve the state of the art on neural program synthesis.

[**R1, R4: Students' concept learning and additional user studies**] We generally agree that more elaborate studies will be needed to fully reach the motivational goal of teaching K-12 students, and evaluate the long term impact on students' concept learning. After the publication of this work, it is quite conceivable to perform larger-scale studies through collaborations with existing educational platforms towards this goal. As suggested by Reviewer 4, we will also consider doing additional user studies varying different components. Nevertheless, the results demonstrate the benefits of our approach.

[**R2, R4: Baseline comparisons for mutation and symbolic execution**] We will include additional ablation studies in the updated paper. As suggested by Reviewer 2, we ran the mutation stage by enumerating the programs within size constraints and then post-checking other constraints without Z3. This implementation leads to a run-time increase by a factor of 10 to 100 for different tasks. So, Z3 seems to be very effective by jointly considering all the constraints.

As a search method, although MCTS seems computationally expensive, the actual run-time and memory footprint of an MCTS run depends on the unique traces explored (i.e., unique symbolic executions done)— this number is typically much lower than the number of iterations, also see Footnote 1 below L571. Furthermore, MCTS is extremely effective as a search strategy, see Fig. 8a. Reviewer 2 asked for a comparison of MCTS with a random search: Considering the MCTS output in Fig. 8c, 8d, to obtain a comparable evaluation score through random search, the corresponding number of unique symbolic executions required is at least 10 times more than executed by MCTS.

[**R1: Related work and clarifications**] We thank the reviewer for pointing out the missing references. Regarding L240–242, we analyzed a random sample of 100 outputs per reference task and we will clarify this in the updated paper. We note that it is possible to increase this number but manually checking properties (V), (VI) is time-consuming for tasks with nested structures. Regarding Delta debugging in L243, we note that the task minimality statistics reported in columns 10, 11 of Fig. 7 are based on human evaluation; Delta debugging was applied primarily to aid the human evaluation. We will add full details in the paper.

[**R2: Multiple I/O pairs and user study results**] Our methodology can be easily extended to multiple I/O pairs for Karel by adapting techniques from Appendix E.2 designed for generating diverse tasks. The reviewer raises an interesting question on the correlation between program control flow and problem concept: This seems to be the case for similar expert-generated problems in platforms like *Code.org* and *CodeHS.com*; also see our response above on additional user studies. SAME's performance is below 1.00 possibly because participants overlooked the solution of Step 1 unaware they will be receiving the same task in Step 2, and the app did not allow them to go back to Step 1.

[**R3: Likert scale for visual dissimilarity**] Indeed, it would be interesting to understand the sensitivity of user study results w.r.t. the scale definition. Another possibility is to use pairwise comparisons in eliciting user evaluations.

[**R4: Clarification on Definition 4**] If a task $\mathtt{T}^{\text{out}}$ can be solved by multiple distinct programs, then all those programs should have the same nesting structure as $\mathtt{C}_{\text{struct}}^{\text{in}}$. This strict definition of conceptual similarity is important as students do not have the solution code $\mathtt{C}^{\text{out}}$. Also, we refer the reviewer to property (V) in L106 and Figure 29 in Appendix E.3.