Chapter 3

(This version is not the final version. It can be some useful material for writing the final report.)

**Summary of previous discussion of whole architecture**

**Distractors**

One of the innovative ideas is to use distractors in an antithetical way. To be more specific, in previous work, distractors are all used to show some incorrect or improper code to let students figure out some fundamental syntax errors and cultivate good coding habits by comparing with the related correct one [1], which is quite beneficial for students’ study in introductory programming. However, when students enter the intermediate level and they have already got familiar with this basic coding knowledge, only setting distractors like the previous work cannot fit them anymore. In this situation, a new idea is picked up - set a group of correct code fragments as distractors instead of discrete incorrect or improper ones. In this way, it is possible to mix different correct algorithms for the same questions in order to let students make out similar solutions. This setting method is significantly applicable in the course Data Structures and Algorithms since there are a lot of questions having this feature – one question with several solving algorithms, for example, bubble sort, selection sort, insertion sort for sorting numbers, and breadth-first search, depth-first-search for searching. Furthermore, it can help students to distinguish similar algorithms in the same categories mentioned before, which is worthwhile when students have learned more than one algorithm and begin to use them motley because of blurry memory. To achieve this function, the project should allow a group of distractors to be input and mixed with the original code fragments randomly. Besides, different versions of solution methods should be allowed to be submitted separately. In other words, the project can allow students to submit multiple times, determine which version the students have submitted, and give feedback to let students know whether they reorder code correctly, they use matched code in one algorithm, or they submit a solution successfully.

**Reference**

[1] Dale Parsons and Patricia Haden. 2006. Parson’s Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE ’06). Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 157–163. http://dl.acm.org.ezproxy.auckland.ac.nz/citation.cfm?id=1151869.1151890