Homework 2 (40 Marks)

Q1. Large language model (LLM) based code completion systems aim to provide an “AI pair programmer”. Concretely, the programmer may provide a code prompt as an input and the pair programmer completes the code. Following is an example of such code completion system. The code outside the box is the code prompt provided by the human programmer and the code inside the box is generated by the AI pair programmer. As shown, the AI pair programmer correctly synthesizes code that searches the key inside the list array.

```
int search_key_in_array (int search_list [], int key) {
    int index;
```

```
    for (index = 0; index < length(search_list); index++) {
      if (search_list[index] == key)
           return index;
     }

     return -1;
}
```

(a) We intend to automatically test/validate the reliability of the AI code completion system. Design a metamorphic test generation framework to test the *correctness* of the code completion system. Concretely, design five metamorphic transformations and their corresponding metamorphic relations. Show your metamorphic transformations with clear examples. For the metamorphic relations, clearly describe how you compare the outputs (i.e., the oracle) of the original and the metamorphically transformed inputs. **Note:** *Each of your metamorphic transformation should be conceptually different. Additionally, your design should be general, not specifically tuned for the example shown above.*
(15 marks)

(b) Assume that you have a set of seed inputs to test the AI code completion system. Based on this assumption, write pseudocode for a randomized test algorithm to metamorphically test the *correctness* of the code completion system. Discuss what seed inputs are needed and your pseudocode should be capable to generate unbounded number of test inputs despite starting with a small finite set of seed inputs. Illustrate first five sample iterations of your test generation in line with the pseudocode.
(10 marks)

(c) Assume that the AI code completion system *mostly* satisfies robustness i.e., *small changes to its input do not affect its output.* Exploit this property to make your metamorphic testing more effective (i.e., finding more erroneous test inputs with respect to the total inputs generated). Modify the pseudocode of the previous question in this respect**.**
(15 marks)