Principles of Programming Languages - Quiz I (Solutions)

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Problems

1. Explain with an example (in **C**), the differences between parameter passing by value and parameter passing by reference.

   Solution: When you pass a parameter by value, only its value is passed to the called procedure. Accordingly, any change made to the parameter by the called procedure is *local* and these changes disappear when the procedure is exited. On the other hand, when a parameter is passed by reference, the address of the parameter is passed to the called procedure. Hence any changes made in the called procedure are permanent (*global*) and will be reflected, even after the procedure is exited. For instance, consider the following codes written in **C** to swap the values of 2 variables, x and y.

```
Function SWAP(int x, y)
1: int temp;
2: temp = x; x = y; y = temp;
```

Algorithm 1.1: Passing through Value

```
Function SWAP(int *x, *y)
1: int temp;
2: temp = *x; *x = *y; *y = temp;
```

Algorithm 1.2: Passing through Reference

   Algorithm (1.1) receives only the values of the parameters of the variables x and y, whereas Algorithm (1.2) receives the addresses of these two variables.

   □

2. Briefly explain the themes of imperative programming, functional programming and logic programming.

   Solution:

   (a) In Imperative programming, e.g., **C**, the programmer directly accesses memory locations through variables and states explicitly what is needed to be done through assignment statements and loops.

   (b) A Functional programming language, e.g., LISP is applicative in nature, since it bases its description of computation on the evaluation of functions. The basic mechanism of such a language is the function call. Ideally, the notions of variables, assignment and loops, do not exist.

   (c) The Logic programming paradigm is based on Symbolic Logic, e.g., PROLOG. The idea here is for the programmer to achieve his end by specifying merely what needs to be done, as opposed to explicitly stating how that thing needs to be done. As in Functional programming, there is no need for control abstractions, such as loops and selection. Such programming is also called "Declarative programming", since it is sufficient to merely declare properties, without specifying the execution sequence.

$\square$

3. What are the different types of errors that can occur in a program? Give one example of each.

   Solution: The different types of errors that can occur in a program are as follows:

   (a) Lexical errors - For instance, a symbol may be used inappropriately, e.g., $x@ = 3$, in **C**, or a keyword may be misspelled.

   (b) Syntax errors - For instance, malformed expressions, missing tokens, etc.

   (c) Semantic errors - Using undeclared variables, making assignments, without respecting types and so on.

   (d) Logical errors - The program logic does not correspond to what the programmer intended.

   $\square$

4. Let $\Sigma = \{0, 1\}$ denote an alphabet and let $L \subseteq \Sigma^*$ denote the language which is composed of strings that start with a 0 and end with a 1. For instance, the strings $\{01, 011101, 0111\}$ belong to L, while the strings $\{\epsilon, 1, 10, 111\}$, do not belong to L. Write a regular expression for L.

   Solution: L is $0 \cdot (0 + 1)^* \cdot 1$ $\square$

5. Let $\Sigma = \{0, 1\}$ denote an alphabet and let $L \subseteq \Sigma^*$ denote the language represented by the following Context-Free Grammar:

$$\begin{aligned} S &\rightarrow A1B \\ A &\rightarrow 0A \mid \epsilon \\ B &\rightarrow 0B \mid 1B \mid \epsilon \end{aligned}$$

   Show that the string 1001 belongs to L, by providing a derivation.

   Solution:

$$\begin{aligned} S &\Rightarrow A1B \\ &\Rightarrow 1B \\ &\Rightarrow 10B \\ &\Rightarrow 100B \\ &\Rightarrow 1001B \\ &\Rightarrow 1001 \end{aligned}$$

   $\square$