EXERCISE #1

One of the motivations for introducing “middleware” is for improving the “portability” of (user) applications. Regarding this, answer the following questions:

#1: What does “(program) portability” mean? Explain the concept, especially by making its difference from “inter-operability” clear.

#2: Show how “middleware” improves “(program) portability using an example (by visualizing the portability”.

EXERCISE #2

Link each proposed (network) programming models/methods (in the left) to its concepts (in the right). Note: the linking can be “many-to-many”.

(1) Thin-client architecture   (a) Software development agility
(2) Fat-client architecture   (b) Integrating legacy programs
(3) Edge-computing           (c) Powerful client-side hardware
(4) ORB                      (d) Easy maintenance
(5) SOA                      (e) Good model for embedded systems
EXERCISE #3

The high-level (network) programming is the counter concept of “the low-level (network) programming. Regarding this concept, answer the following questions:

#1: What is “high-level (network) programming” (please make its concept clear by making clear contrast against “low-level (network) programming)?

I suggest you to answer this question using some technical terms, which are supposed to be used (especially by any CS major students for describing what is “high-level programming”).

#2: Mention two examples for each of “low-level network programming” and “high-level network programming”.

#3: What are the primary advantages of “low-level network programming” compared with and “high-level network programming” (mention at least two).

What are the primary disadvantages of “low-level network programming” compared with and “high-level network programming” (mention at least two).

EXERCISE #4

About “History of network application architectures”:

Why “mainframe & dumb client” for the first generation?

EXERCISE #5

About “History of network application architectures”:

Why “HTML client for web applications” for the third generation?