Software Fundamentals – 4%

Domain Scope

1. Skills and fundamental programming concepts, data structures, and algorithmic processes
2. Programming strategies and practices for efficient problem solving
3. Programming paradigms to solve a variety of programming problems.

Domain Competencies

A. Justify the way IT systems within an organization can represent stakeholders using different architectures and the ways these architectures relate to a system lifecycle. *(Requirements and development)*
B. Demonstrate a procurement process for software and hardware acquisition and explain the procedures one might use for testing the critical issues that could affect IT system performance. *(Testing and performance)*
C. Evaluate integration choices for middleware platforms and demonstrate how these choices affect testing and evaluation within the development of an IT system. *(Integration and evaluation)*
D. Use knowledge of information technology and sensitivity to the goals and constraints of the organization to develop and monitor effective and appropriate system administration policies within a government environment. *(System governance)*
E. Develop and implement procedures and employ technologies to achieve administrative policies within a corporate environment. *(Operational activities)*
F. Organize personnel and information technology resources into appropriate administrative domains in a technical center. *(Operational domains)*
G. Use appropriate and emerging technologies to improve performance of systems and discover the cause of performance problems in a system. *(Performance analysis)*

Software Fundamentals Subdomains

01 Perspectives and impact
(Level 1 minimal degree of engagement)
Competencies:

a. Reflect on how the creation of software has changed our lives.
b. Synthesize how software has helped people, organizations, and society to solve problems.
c. Describe several ways in which software has created new knowledge.

02 Concepts and techniques
(Level 2 medium degree of engagement)
Competencies:

a. Compare multiple levels of abstraction to write programs (constants, expressions, statements, procedures, parameterization, and libraries).
b. Select appropriate built-in data types and library data structures (abstract data types) to model, represent, and process program data.
c. Use procedures and parameterization to reduce the complexity of writing and maintaining programs and to generalize solutions.
d. Explain multiple levels of hardware architecture abstractions (processor, special purpose cards, memory organization, and storage) and software abstractions (source code, integrated components, running processes) involved in developing complex programs.
e. Create new programs by modifying and combining existing programs.

03 Problem-solving strategies
(Level 1 minimal degree of engagement)
Competencies:

a. Explain abstractions used to represent digital data.
b. Develop abstractions when writing a program or an IT artifact.
c. Apply decomposition strategy to design a solution to a complex problem.
d. Explain appropriateness of iterative and recursive problem solutions.
e. Write programs that use iterative and recursive techniques to solve computational problems.

04 Program development
(Level 3 large degree of engagement)
Competencies:

a. Develop a correct program to solve problems by using an iterative process, documentation of program components, and consultation with program users.
b. Use appropriate abstractions to facilitate writing programs: collections, procedures, application programming interfaces, and libraries.
c. Evaluate how a program is written in terms of program style, intended behavior on specific inputs, correctness of program components, and descriptions of program functionality.
d. Develop a program by using tools relevant to current industry practices: version control, project hosting, and deployment services.
e. Demonstrate collaboration strategies that consider multiple perspectives, diverse talents, and sociocultural experiences.

05 Fundamental data structures
(Level 2 medium degree of engagement)
Competencies:

a. Write programs that use data structures (built-in, library, and programmer-defined): strings, lists, and maps.
b. Analyze the performance of different implementations of data structures.
c. Decide on appropriate data structures for modeling a given problem.
d. Explain appropriateness of selected data structures.

06 Algorithm principles and development
(Level 2 medium degree of engagement)
Competencies:

a. Describe why and how algorithms solve computational problems.
b. Create algorithms to solve a computational problem.
c. Explain how programs implement algorithms in terms of instruction processing, program execution, and running processes.
d. Apply appropriate mathematical concepts in programming: expressions, abstract data types, recurrence relations, and formal reasoning on algorithm's efficiency and correctness.
e. Evaluate empirically the efficiency of an algorithm.

07 Modern app programming practices
(Level 1 minimal degree of engagement)
Competencies:

a. Create web and mobile apps with effective interfaces that respond to events generated by rich user interactions, sensors, and other capabilities of the computing device.
b. Analyze usability, functionality, and suitability of an app program.
c. Collaborate in the creation of interesting and relevant apps.
d. Build and debug app programs using standard libraries, unit testing tools, and debuggers.
e. Evaluate readability and clarity of app programs based on program style, documentation, pre- and post-conditions, and procedural abstractions.

Note: Level L1 (L1) used within a subdomain indicates a minimal degree of engagement associated with the learning proficiency of the fundamentals of the subdomain.

Levels 2 (L2) and 3 (L3) used within a subdomain indicate medium and large degrees of learning engagement associated with the application and transferring of learning to complex problems and situations.