Design Concepts – Q&A

1. Provide examples of three data abstractions and the procedural abstractions that can be used to manipulate them.

2. Describe software architecture in your own words.
   Software Architecture is the structure or organization of program components (modules), the manner in which these components interact, and the structure of data that are used by the components. In a broader sense, however, components can be generalized to represent major system elements and their interactions.

3. Describe separation of concerns in your own words. Is there a case when a divide-and conquer strategy may not be appropriate? How might such a case affect the argument for modularity?
   Separation of concerns involves solving a complex problem by breaking it up into separately solved subproblems. There are cases in which different parts of a problem are interrelated in a manner that makes separate considerations more complex than combined considerations. Highly coupled problems exhibit this characteristic. However, continuing combination of problem parts cannot go on indefinitely because the amount of information exceeds one's ability to understand. Therefore, when (9.2) is not true, modularity may be modified, but not eliminated

4. When should a modular design be implemented as monolithic software? How can this beaccomplished? Is performance the only justification for implementation of monolithic software?
   In some time critical applications, monolithic implementation may be required. However, design can and should occur as if the software was to be implemented modularly. Then "modules" are coded in-line.

5. Apply a "stepwise refinement approach" to develop three different levels of procedural abstractions for one or more of the following programs:
   (a) Develop a check writer that, given a numeric dollar amount, will print the amount in words normally required on a check.
   (b) Iteratively solve for the roots of a transcendental equation.
   (c) Develop a simple task scheduling algorithm for an operating system.

We create a functional hierarchy and as a result refine the problem. For example, considering the check writer, we might write:

Refinement 1:

```
Write dollar amount in words
```

Refinement 2:

```
Procedure write amount;
    Validate amount is within bounds;
    Parse to determine each dollar unit;
```

```
            Generate alpha representation;

        end write_amount

Refinement 3:

        procedure write_amount;

            do while checks remain to be printed

                if dollar amount > upper amount bound

                    then print "amount too large error message;

                else set process flag true;

                endif;

                determine maximum significant digit;

                do while (process flag true and significant digits remain)

                    set for corresponded alpha phrase;

                    divide to determine whole number  value;

                    concatenate partial alpha string;

                    reduce significant digit count by one;

                enddo

                print alpha string;

            enddo

      end write_amount
```

6. Briefly describe each of the four elements of the design model.
   - Data/Class design - created by transforming the analysis model class-based elements into classes and data structures required to implement the software
   - Architectural design - defines the relationships among the major structural elements of the software,
   - Interface design - describes how the software elements, hardware elements, and end-users communicate with one another
   - Component-level design - created by transforming the structural elements defined by the software architecture into a procedural description of the software components

7. List three characteristics that can serve as a guide to evaluate design quality.

- Design implements all explicit requirements from the analysis model, as well as accommodating implicit customer requirements.

- Design must be understandable to the people who generate the code to implement design, those who test it, and those who support it.

- Design must provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

8. Explain how effective modular design is achieved through functional independence of the individual modules?

Functional independence of modules is achieved by making modules single-minded (high cohesion) and preventing excessive interaction (low coupling) with other modules or system elements. Independent modules are easier to develop, maintain, and test, because the impact of side effects is reduced (as is the propagation of errors). This also makes it easier to perform parallel implementation of modules.

9. Describe the principle of information hiding as it applies to software design.

The principle of information hiding implies that modules only share information with each other on a "need to know" basis to achieve some specific software function. Hiding enforces the procedural constraints to both the module procedural detail and any data structures local to the module.