**ISHOLA ABDULMALIK OLATUNDE**

**18/SCI01/040**

**CSC206**

REVISED QUESTIONS PART 1

1i.Computer programming is the process of designing and building an executable computer program to accomplish a specific computing result.

1ii.A computer program is a collection of instructions that can be executed by a computer to perform a specific task. A computer program is usually written by a computer programmer in a programming language.

1iii.A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms.

1. **Machine language** is a collection of binary digits or bits that the computer reads and interprets.An **assembly language** is a **[programming language](https://simple.wikipedia.org/wiki/Programming_language%22%20%5Co%20%22Programming%20language)** that can be used to directly tell the computer what to do. An assembly language is almost exactly like the [machine code](https://simple.wikipedia.org/wiki/Machine_code%22%20%5Co%20%22Machine%20code) that a computer can understand, except that it uses words in place of numbers.A high-level language is a [programming language](https://techterms.com/definition/programming_language) designed to simplify computer programming. It is "high-level" since it is several steps removed from the actual code run on a computer's [processor](https://techterms.com/definition/processor).
2. Simplicity

.Readability

.Naturalness

.Abstraction

.Efficiency

.Structured

.Compactness

.Locality

1. **Procedural Programming**: Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have low level of complexity

**Object-oriented Programming**: Here the solution revolves around entities or objects that are part of problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

**Functional Programming**:Here the problem, or the desired solution, is broken down into functional units. Each unit performs its own task and is self-sufficient. These units are then stitched together to form the complete solution.

**Logical Programming**: Here the problem is broken down into logical units rather than functional units.

**Top-down or ModularApproach:** The problem is broken down into smaller units, which may be further broken down into even smaller units. Each unit is called a module. Each module is a self-sufficient unit that has everything necessary to perform its task.

**Bottom-upApproac**h :In bottom-up approach, system design starts with the lowest level of components, which are then interconnected to get higher level components. This process continues till a hierarchy of all system components is generated. However, in real-life scenario it is very difficult to know all lowest level components at the outset. So bottoms up approach is used only for very simple problems.

1. Defining the problem

Planning the solution

Coding the program

Testing the program

Documenting the program

1. **KISS:**Nobody in programming loves to debug, maintain, or make changes in complex code. “**Keep It Simple, Stupid (KISS)**“ states that most systems work best if they are kept simple rather than making it complex, so when you are writing code your solution should not be complicated that takes a lot of time and effort to understand.

**DRY:**Duplication of data, logic, or function in code not only makes your code lengthy but also wastes a lot of time when it comes to maintaining, debug or modify the code. If you need to make a small change in your code then you need to do it at several places. **“Don’t Repeat Yourself (DRY)”** principal goal is to reduce the repetition of code.

**YAGNI:**Your software or program can become larger and complex if you are writing some code which you may need in the future but not at the moment. **“You Aren’t Gonna Need It (YAGNI)”** principle states that “don’t implement something until it is necessary” because in most of the cases you are not going to use that piece of code in future.

**SOLID:**The SOLID principle stands for five principles which are Single responsibility, Open-closed, Liskov substitution, Interface Segregation, and Dependency inversion.

1. Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection and repetition, block structures, and subroutines.
2. It encourages top-down implementation, which improves both readability and maintainability of code.

It promotes code reuse, since even internal modules can be extracted and made independent, residents in libraries, described in directories and referenced by many other applications.

It's widely agreed that development time and code quality are improved through structured programming.