**FAQ on Programming Foundation with Pseudo code**

|  |  |
| --- | --- |
| **Q. 1** | **What are good Programming Practices?** |
| Ans. | * Keywords should be ideally capitalized while writing pseudo code * Comments should be written above the module/ pseudo code. * Follow characteristics of a good program like readability, layout, meaningful variables and module names, modularity, comments |
| **Q. 2** | **What is maintainability?** |
| Ans. | * If the program is easy to understand and If it is easy to modify then the program is called as maintainable * Achieve maintainability by eliminating hard coded constants from the code. |
| **Q. 3** | **What is modularity? What are the benefits of Modularity?** |
| Ans. | * A small unit of code for a single purpose is called module. * Module can be a function, a method, a procedure or a sub-program or a component. * Modularity is dividing the entire program into modules.   **Benefits**   * Reduce complexity : By using the abstracting power of modules, complex code   can be made to appear simpler and easy to understand   * Better documentation : By putting a set code into a well-defined module, makes   the code self –explanatory   * Reuse code: Once a module is written, the same module can be reused in another application.   • Easy to incorporate changes, as required so that the code will be more maintainable. |
| **Q. 4** | **What are the Characteristics of well-defined modules?** |
| Ans. | * They perform a single well defined functionality * High Cohesion – do one thing, and do it well * Low Coupling – reduce dependencies between modules * Meaningful names – Be specific (Ex: ReadRecord) * Parameter passing must be accurate and there is no type mismatch for any parameter   ***c***alculateTotal(Integer price, Integer quantity)  Refer the valid and invalid statements to invoke a module  calculateTotal(3,5); //Valid  calculateTotal(4,3,4); //Invalid  calculateTotal(‘Test’,3); //Invalid |
| **Q. 5** | **What is coupling? Explain the difference between tight coupling and loose coupling?** |
| Ans. | Coupling or Dependency is the degree to which each program module relies on  each other.   * Tight or High Coupling – modules are greatly dependent on each other * Loose or Low Coupling – reduce dependencies between modules * Passing more parameters means high coupling. We should strive for low coupling |
| **Q. 6** | **What is cohesion? What is the importance of cohesion?** |
| Ans. | Cohesion is a measure of how the activities within a single module are related to one another.   * High Cohesion – do one thing, and do it well * Writing function with high cohesion is always good practice. * Ideally, we should strive for High Cohesion and Low Coupling. |
| **Q. 7** | **What is the difference between correctness and robustness?** |
| Ans. | * Correctness means building code which never returns inaccurate results * Safety critical applications tend to focus on correctness, failure to achieve a result being regarded as better than inaccurate result. * Robustness favor’s the return of any result, even the inaccurate one. * Consumer applications typically favor robustness as any result is better   than software crashing |
| **Q. 8** | **What is defensive programming and its purpose?** |
| Ans. | * Defensive programming enables us to detect minor problems early on, rather than get bitten by them later when they’ve escalated into major disasters. * Defensive programming is a method of prevention, rather than a form of cure.   **Purpose of defensive programming**   * Ensure that a program never returns inaccurate result even though valid data is passed. * Abnormal termination of the program will be avoided * Even though, invalid data is passed instead of abnormal termination of program execution, meaningful error messages has to be displayed. * Thinking all the possible and impossible scenarios and take care of exception to prevent problem occurrence. |
| **Q. 9** | **What are different techniques of defensive programming** |
| Ans. | **1. Input validation :**   * Check the values of all data from external sources * Validate the data exchanged between modules * Decide how to handle bad inputs * Validate the data for consistency * Validate the data type * Validate the data range   **2. Error handling :**   * This techniques deal with errors you would expect to occur in code   **3. Error Containment :**   * Protect your code from an invalid data coming from “outside”( Data from an external system) * Establish “barricades”(module where validation logic exists) for leaving dangerous data outside of the boundary(like before invoking module) |
| **Q. 10** | **What is exception handling?** |
| Ans. | * An exception is an event that occurs during the execution of a program that disrupt its normal course. * No matter how well-designed a program is, there is always a chance that some kind of error will arise during its execution * Unexpected events happen and the programmer should always be prepared for the worst. * **Examples**: Hard disk crash; Out of bounds array access; Divide by zero |
| **Q. 11** | **Guidelines for creating exception handlers. Which module should throw exception? Which module should catch exception?** |
| Ans. | * We need to think of all possible conditions from the case study that are likely, unlikely, and impossible. * We need to analyze the consequences of failures along with the probability. * Accordingly create an exception for each of the possibility * Document the reason for the exception * **Raise the exception** whenever the condition arises in the sub module * **Catch the exception** in the appropriate parent module |
| **Q. 12** | **What is the definition and purpose of testing and debugging?** |
| Ans. | * Testing is the process of executing a program with the intent of finding errors * Testing is a process used to help identify the correctness, completeness and quality of a developed computer software * Debugging is an art used to “isolate”, and “correct” the cause of an error * Debugging is performed by developers to uncover where a defect in the code exists and correct it |
| **Q. 13** | **What is static and dynamic testing? What are the techniques of static and dynamic testing?** |
| Ans. | **Static Testing:** Testing a software without execution on a computer   * + Review : Review the created artifacts using checklist   + Code Inspection : Code inspection is a set of procedures and error detection techniques for group code reading.   + Walkthrough : Like code inspection it is also an group activity.   **Dynamic Testing** : Testing a software by execution using sample input values.   * + White Box testing : Used to test the internal structure of the code   + Black Box Testing : Test the functionality of application by providing input and getting expected output |
| **Q. 14** | **What are the guidelines for implementing test cases** |
| Ans. | * Write test case for all the requirements specified in the application * Take care of writing test case for non-functional requirements like security, performance, etc.. * If any test case fails, log the failed test cases as defect in defect tracking sheet. * Check for all boundary conditions. * **80-20 Rule:** In most systems, 20% of the modules account for 80% of the defects found. The probability of finding defect in a module is directly proportional to the number of defects already found in the module. * Do self-review and peer review for all test cases as quality of test case affects testing. |
| **Q. 15** | **What is system testing, validation testing, acceptance and regression testing?** |
| Ans. | **System Testing :** a complete integrated system as a whole, in order to evaluate compliance with respect to specified requirements  **Validation Testing :** It checks whether the program matches it’s external specifications and to have a final check to see whether it is indeed the right product  **Acceptance Testing :** Acceptance Testing focuses on testing whether the right system has been created. It is usually carried out by the end user  **Regression Testing :** Regression Testing involves “selective re-testing” of the system or it’s components after the changes are done. It is done to verify absence of unintended effects and to verify compliance with all (old and new) requirements. |