SAMPLE SOLUTIONS

Q1. What is a *non-functional requirement*? Give two (2) examples to illustrate your answer.

A NFR is a statement of a constraint or expected behaviour that applies to a system.
Ex1: The system must respond to user requests within 1 minute
Ex 2: The system must be secure from unauthorised access to system data.
... and many more examples

Q2. Give an example of an *unverifiable* requirement, and then rewrite that requirement so that it is verifiable.

R1. The system must be user friendly
R1 is not verifiable because it depends on the users, their experience etc.
R2. After a one hour training system, 90% of users must be able to complete successfully the following scenario using the system … detail the scenarios here.

Q3. Why is it important to agree the *priority* of different requirements in a software engineering project?

• So developers can implement the most important requirements first, with least important ones being left out or delayed to future releases.
• Helps with negotiation between stakeholders, so that all stakeholders are happy with the product.
• Helps focus attention on the main goals of the project, and how each requirement contributes to that goal.

Q4. What is a *function point*? Give one advantage and one disadvantage of function points.

• A FP is a measure for the size of a system in terms of the functionality of that system.
• It is calculated from the inputs, outputs and processing components of the requirements.
• Advantage: can be calculated at the beginning of the project and so used for estimation
• Disadvantage: subjective and takes training to measure. Not very accurate for estimation.

Q5. Estimating the *time* required to develop a software system is difficult. How might you reduce the error of such an estimate at the beginning of a project?

• Getting several opinions (Delphi approach) can help reduce error in initial estimates.
• Use more than one estimation method, and more than one estimator, to get the spread of possible estimates.
• Then use expert opinion to come to an agreed estimate based on the range of initial ones.

Q6. A company wishes to evaluate the quality of the user documentation it provides with its products. Propose two (2) ways they could measure the *quality* of user documentation, giving a suitable *metric* (measurement unit) for each.

Metric 1: measure user satisfaction using a survey rating different features of the documentation, say on a scale of 0 (very dissatisfied) to 5 (excellent) and then average the responses to get a measure of satisfaction
Metric 2: set some user tests: can you find out how to use the system for the following scenario (eg. login, enrol in unit, change something …)
... And also many more including
Metric 3: count how many help screens or pop-ups are available in the system (good for quantity, but doesn’t capture the quality of the help given)