What are the two reasons for which the software engineering is important?

Software engineering is an engineering discipline that is concerned with all aspects of software product from the early stages of system specification through to maintaining the system after it has gone into use. Software engineering is important for two reasons:
1. More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
2. It is usually cheaper, in the long run, to use software engineering methods and techniques for professional software systems rather than just write programs as a personal programming project. Failure to use software engineering method leads to higher costs for testing, quality assurance, and long-term maintenance.

Discuss whether professional engineers should be certified in the same way as doctors or lawyers.

These are possible discussion points - any discussion on this will tend to be wide ranging and touch on other issues such as the nature of professionalism, etc.

Advantages of certification

• Certification is a signal to employers of some minimum level of competence.
• Certification improves the public image of the profession.
• Certification generally means establishing and checking educational standards and is therefore a mechanism for ensuring course quality.
• Certification implies responsibility in the event of disputes. Certifying body is likely to be accepted at a national and international level as 'speaking for the profession'.
• Certification may increase the status of software engineers and attract particularly able people into the profession.

Disadvantages of certification

• Certification tends to lead to protectionism where certified members tend not to protect others from criticism.
• Certification does not guarantee competence merely that a minimum standard was reached at the time of certification.
• Certification is expensive and will increase costs to individuals and organisations.
• Certification tends to stultify change. This is a particular problem in an area where technology developments are very rapid.

Why are evolutionary models considered by many to be the best approach to software development in a modern context?

Because time lines for the development of modern software are getting shorter and shorter, customers are becoming more diverse (making the understanding of requirements even harder), and changes to requirements are becoming even more common (before delivery), we need a way to provide incremental or evolutionary delivery. The evolutionary process accommodates uncertainty better than most process models, allows the delivery of partial solutions in an orderly and planned manner, and most importantly, reflects what really happens when complex systems are built.

Explain what is wrong with the notion that computer software does not need to evolve over time.

Computer software must be revised as errors are discovered and corrected. Software must be updated to accommodate changes in the computing environment. Many times a customer will request changes to add new functions to an existing product or to accommodate changes in the business environment. Sometimes an older system will need to be reengineered to provide benefits to the user in a modern context. The bottom line is that software that does not evolve will eventually become unusable.

How do software characteristics differ from hardware characteristics?

Software is developed, not manufactured. Software does not wear out. Most software is custom built, not assembled out of components.

How does software differ from the artifacts produced by other engineering disciplines?

Software is both a product and a vehicle for delivering a product. As a product, software is an information transformer. As a vehicle for delivering a product, software serves as a basis for computer control, communication, and creation of other programs.