Task 1:

"No Silver Bullet " is a widespread paper in software engineering written by F. Brooks. By this paper he has brought the expression ”Silver Bullet” to the software engineering field. The main argument of the paper is: *" There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement in productivity, in reliability, in simplicity"* and this non-existent development technique is the silver bullet. He didn't expect any existence for any silver bullet that could increase the productivity of the software development process for at least ten years.
He compared the software to the hardware technology progress which is advanced quickly and become cheaper. In the abstract he gives four suggestions that are used nowadays by the IT companies: 1- Buy software that could be bought instead of developing it, 2- Using prototypes to define the software requirements, 3- Growing software organically, 4- Identifying and developing the great conceptual designers of the rising generation.
Brooks divides the software technology difficulties into two kinds: Essential difficulties those are the crucial issues caused by the nature of the software and accidental difficulties those are not inherent to the process and caused by the way of building the software. He lists four essential difficulties to software development: complexity, conformity, changeability, and invisibility. Here I will encounter three of them:

- **Complexity**: The software is a collection of components or data structures interact with each other which are not identical and different from a software to another. By the process of building the software, the number of elements increase. The complexity of the software increases much worse than linearly with its size. Complexity causes technical problems like the difficulty of communication among team members, much less understanding, all the possible states of the software and the difficulty of extending programs to new functions without creating side effects. Besides, complexity causes management problems like the difficulty of the personnel turnover because of enormous learning and understanding load.
- **Changeability:** All successful software changes frequently. The changes in the software produce by applications, users, machines, laws, and hardware problems which always are subjected to changes. Even if the change in software is easier to compare with hardware, the complexity of the system becomes much harder. The changeability is an extremely substantial characteristic of a good and a success software.
- **Invisibility:** We cannot visualize all portions of the software at once. Software is unvisualizeable and has no ready geometric representation as the mechanical parts have scale drawing and the buildings have floor plans. Engineers start to develop the software without having any clear idea about the final product since the complete view is incomprehensible. They attempt to draw diagrams for the software design and structure, and this eliminates the complexity in one way or another. The problem that still exist is that each person can see or draw the diagram in a different way which doesn’t only hinder the understanding within one mind, it makes the communication among minds harder.

Brooks mentions three solutions to the accidental difficulties as past breakthroughs which are the high-level languages, time-sharing, and the unified programming environments

In contrast, he didn’t assert the lack of solutions to the essential difficulties. He suggests solutions that could be the silver bullet in the future which could kill the software productivity monster. He gives several solutions that mitigate the essential difficulties like using object-oriented (OO) programming, artificial intelligence, expert systems, automatic programming, graphical programming, program verification, environments and tools, and workstations.