Exercise 11:

### ****Structured Programming****

#### Advantages:

* **Improved code clarity**: Breaking down a program into modules or blocks improves readability.
* **Easier to debug**: Smaller sections of code (modules) make it easier to find errors.
* **Enhanced maintainability**: Changes to individual modules don’t affect the entire program.
* **Modularity**: Encourages the reuse of code by dividing it into independent modules.

#### Disadvantages:

* **Limited scalability**: As programs grow large, structured programming can become less manageable.
* **Lack of abstraction**: Does not naturally support concepts like inheritance or polymorphism, making it harder to reuse code in more complex systems.

### ****Procedural Programming****

#### Advantages:

* **Simplicity**: Easy to understand, especially for small programs.
* **Reusability**: Functions can be reused across the program or even other programs.
* **Debugging**: Isolating logic within procedures simplifies debugging and testing.

#### Disadvantages:

* **Tight coupling**: Procedures often rely on shared data or global variables, making maintenance more difficult as the program grows.
* **Lack of data security**: No inherent mechanism to restrict access to data, as all parts of the program can manipulate global variables.
* **Scalability issues**: Not ideal for very large programs, as functions may become too intertwined and difficult to manage.

### ****Object-Oriented Programming (OOP)****

#### Advantages:

* **Modularity**: Code is organized into self-contained objects, making it easier to manage, extend, and maintain.
* **Reusability**: Through inheritance, common functionality can be shared among classes, reducing code duplication.
* **Scalability**: Large and complex systems are easier to manage, as new features can be added with minimal changes to existing code.
* **Data protection**: Encapsulation hides internal object states, preventing unwanted data manipulation.

#### Disadvantages:

* **Complexity**: OOP can introduce unnecessary complexity for smaller programs, where simpler paradigms like procedural programming might be more efficient.
* **Performance overhead**: Object management, memory consumption, and additional layers of abstraction can slow down execution compared to procedural programs.
* **Steeper learning curve**: Concepts like inheritance, polymorphism, and abstraction can be challenging for beginners to grasp.