## Task 1 – Understanding Procedural Programming

#### **Scenario:** You work for a software development company, Financial IT Solutions Ltd, who have been recently asked by a client to develop a payroll processing application.

#### Your manager is considering utilising the procedural programming paradigm for the development of the payroll application. You have been asked to produce a report containing between 500 and 1,000 words that explores the procedural programming paradigm and its advantages and disadvantages. In addition to discussing the paradigm itself, summarise the types of projects where it is best suited.

### What is Procedural Programming

Procedural Programming is a paradigm of programming that uses sequential functionality as the main code implementation. This functionality is a sequence of instructions called procedures or functions. In Procedural Programming, we split the code into smaller blocks and implement it as a function. Each function is assigned a specific task and can be reusable multiple times.

**Top-Down Approach:** Procedural Programming is codding in a top-down approach, smaller blocks of code that start from the top and go down by the code. Also, the Top-Down approach is related to the scope.

**Parameter passing:** The functions could receive parameters and return values, so the functionality of functions very often is to receive and return data. Different types of functions can only receive and execute the code, for example, to display information, or calculate and update data in a database. Or as an instance, a function could ask for user input and return an answer for further processing. It’s strongly recommended to use this approach (passing and receiving parameters) for function in procedural programming as it’s always easy to read, understand and use.

**Scope:** Scope is different access layers in Procedural Programming that outline the usage of the functions and variables. For example, if we declare the function inside the main, we can call it only from the main and this function can access all the main variables, but outside of the main, it is not possible to call or access it without globalizing. As Procedural Programming is the top-down access, the variables are only visible in sub-functions, but not that are above our function hierarchy.

**Local variables:** A variable is some allocation of the memory in a computer. When we declare the variables inside the function, they become local variables, which means that they exist only in the scope of this function and when the function is finished, the local variables will be destroyed. It’s vital to pay attention to returning processed information before it is destroyed by a built-in garbage collector. When variables as parameters they also become local copies of those variables, so changes made to these variables won’t be done to the main ones, if the variables are not returned and reassigned to the main ones.

**Global variables:** Other functions could not receive the data directly, but use global variables and work with them. In Python, the global variables are very complicated because they are stored as an object and to make some changes to it, we should only interact with them, but reassigning new values, for example, a recreated dictionary, won’t change the main object.

Global variables should be used in very limited circumstances and could be justified. Also, it’s important to consider that we are able only to change the data inside the global variables, but not redirect them to another memory allocation directly.

**Global Variables in Python:** One of the options that is available in Python is globalising variables inside the function. Then in the program score, those variables receive higher hierarchical status and could be visible in the program score, but only if the function was executed.

**Modularity:** Modular coding is a process that divides a complex task into many smaller tasks and runs those tasks in a sequence to benefit the bigger task.

### Advantages:

* **Simplicity:** Procedural Programming is very easy to understand and code due to splitting the code structure into smaller blocks.
* **Reusable code:** The code can be reused multiple times reducing code lines can by simply repeating those functions. Procedural Programming follows the main DRY principle, reuse code where it’s possible to.
* **Easy to test:** The smaller blocks of code, and function, are much easier to test and debugging can be done with some sample parameters.
* **Team-Friendly:** Each programmer could work on their function in a bigger project, so each person is responsible for the
* **Accessibility:** It’s easy to find information about procedures and learn programming in this way.
* **Smaller Memory Requirements:** As functions use mostly local variables and save by smaller blocks of memory. It becomes very useful if the database is large and complex, using procedural programming will require much less memory because it will process the data in smaller segments, for which Procedural Programming is not recommended.
* **Best suitable projects:** Procedural Programming is perfect for smaller projects and general programming.

### Disadvantages:

* **Limited support for Abstraction:** Even though Procedural Programming brings some abstraction through functions and procedures, it is inferior to other paradigms such as Object-Oriented Programming, which usually has much better abstraction support.
* **No Portability:** In Procedural Programming, code can be reused within a single project, but it cannot be exported to other projects without adjustments. So we need to rewrite most of the code when working on other projects.
* **Data Security:** Data is exposed and accessible to multiple procedures in Procedural Programming. It is not suitable for projects where data protection is the priority.
* **Global data and side effects:** Procedural programming often makes use of global data and variables that can lead to side effects, making it more difficult to justify the code functionality and maintain its integrity.
* **Complex, large-scale projects:** Within project growth, managing code separation of functionality and data sharing between functions in Procedural Programming becomes challenging. This paradigm is not suitable for scalable projects.