Assignment 1:

Part 1:

1.

Imperative programming is a paradigm which based on well-defined sequence on instructions to the computer.
Meaning, a series of commands, which specify what the computer has to do – and when – in order to achieve a desired result.

Procedural programming is similar to the imperative programming by involving writing down the list of instructions to tell the computer what to do step by step, however, this paradigm treats data and procedures as two different entities.

Functional programming is a paradigm that constructs the code using functions in order to reach the desired goal.
meaning Functions are the fundamental to organize the code.

- The procedural paradigm improves over the imperative by simplifying and organizing the code; separating the variables declaration and the sequence of instructions.
- The functional paradigm improves over the procedural paradigm by preventing double coding, allowing using recursions and unlike procedural programming there are no side effects.

2.

```
function averageGradesOver60(grades: number[]){
    const result = grades.filter(over60);
    const output = result.reduce(sum);
    return output/result.length;
}

function over60(num: number){
    return num > 60;
}

function sum(num1: number, num2:number){
    return num1+num2;
}
```

3.

a. <**T**> (x : **T[] ,** y : (a: **T**) => **boolean**) => x.some(y) : **boolean** .

b. x: **number[]** => x.reduce((acc: **number**, cur: **number**) =>
=> (acc + cur): **number**, 0): **number** .

c. <**T**> (x : **boolean**, y : **T[]**) => x ? y[0] : y[1] :**T** .

d. <**T**> (f: (a : **T**) => **T**, g: (a : **T**) => **T**) => x : **number** => f (g(x+1)) : **T** .
*the type of the argument of f is the type of the output of g.

4.

Abstraction Barriers is a concept in programming in which the programmer, instead of making complex operations inline, separates those operations into several functions, and those functions are the structure of the code.
Thus, she creates a barrier, where there is no need to think about the internals of how the complex operation gets calculated.
In this concept, the programmer usually will use a meaningful name for those functions.

Eventually, instead of having a clumsy algorithm with many code lines, we get an abstract, cleaner code, which is much easier to read.