PPL Assignment 1 Part 1

Question 1:

Imperative- Imperative is a programming paradigm which describes explicitly how the program operates. It is basically an explicit sequence of commands where the commands changes the state of program.

Procedural- Procedural is a programming paradigm derived from imperative programming which organized with hierarchies of procedure calls. A procedure is a small unit of commands that interacts with interfaces published by other procedures and its own local variables, without affecting the state of the program outside the scope. Procedural paradigm encourages modular programming which separates the functionality of the program into independent modules.

Functional- Functional is a declarative programming paradigm, where programs are constructed by applying and composing functions, which avoid any global state mutation. Here the view the program as an expression that can be evaluated from successive functions applied on arguments, after substituting the result from the functional expression.

Procedural programming improve over imperative paradigm by using procedures. As mentioned above, procedure is a small unit of commands with well-defined interface and expected behavior. Each procedure uses structures such as variables, array data structure and loop control structure, it uses them locally without affecting the state of the program outside the scope. The advantages of this paradigm are reflected by: reduce of code repetition, easily adapted to different values of the parameters, the coupling between the code parameters is clearer and constructed inside a block of statements.

Functional programming improve over procedural programming by having no side-effects. That is, in contrast to procedural programming, it doesn't mutates the parameters. It makes it easier to perform concurrent programming since data should be immutable.

Question 2

a- <T>(x:T[], y: ((z: T) => boolean)) => boolean;

b- (x: number[]) => number;

c- <T>(x:boolean, y: T[]) => T | undefined (undefined for the case where y.length < 2)

Question 3

Abstraction barrier is an idea in which we create a level of abstraction between collections of procedures, such that high level procedures can only call lower level procedures. A high level procedure doesn't need to know the details of how a lower level procedure works, it can use this procedure's reliability as some kind of black box that implements the lower abstraction data. So the procedures at each level are the interfaces that define the abstraction barrier and connect the different levels. It makes the program easier to maintain and modify, also it allows us to represent a complex object in a variety of ways with lower level abstraction procedures.