PPL - Assignment 1

Part 1: Theoretical Questions

1. Dimensions of variability across programming paradigms

Paradigms are distinguished by programming practices they encourage and forbid, Here are few dimensions to which the paradigms differ from one another:

Control Flow: how execution flows within the program, these are some different control flow styles:

- Sequential programming
- Branching: we can either go one way or another, usually taking place when asking something, for example in 'if' and 'case' functions
- Concurrent threads: executing few parts of the code at the same time
- Reactive manner: the code executes when a new event is taking place
- Declarative: declare what we want to get as an answer and not explicitly saying how the program flow will be executed, for example: in SQL and prolog

Syntax: how natural, brief, readable is the expression of code given the syntax of the language. Can the syntax of language be extended by the programmer. Some languages are easier for writing and reading and some not, for example:

- SQL is a verbose language which is a lot similar to English sentences
- Scheme have very few keywords and many parentheses.

Domain: to which application domain is the paradigm best applicable, for example:

- C or Java are more efficient and scalable for server-side processing
- Javascript usually good for user interface on the web and for different front-end applications
- SQL used for databases

2. Types of Functions

a. (x: number, y: number) => number

b. <T>(arr: T[]) => T

c. (x: boolean, y: number) => number

3. Shortcut semantics

Shortcut Semantics is a concept where the next operand is evaluated only when the result is not fully determined by the previous operand. At the moment the result is determined, the evaluation stops and a value is returned. Both, shortcut semantics and non-shortcut semantics return the same values, therefore it is difficult to distinguish between them. One way for distinguishing between the two is by using deliberate error throwing.

We can take for example the native 'some' and 'every' methods which employ the shortcut semantics concept. The method 'some' stops and immediately returns true at the moment it finds an element that satisfies the predicate. The method 'every' stops and immediately returns false at the moment it finds an element that does not satisfy the predicate. Also, In the Practical Session we saw an example to distinguish between the shortcut concept and the non-shorted one implementing on the 'some' method, by triggering an error, using the throw primitive with try/catch.