By cijhho123 and T.K.

Part 1

Question 1

(a)

1) Imperative programming: A software paradigm in which the program consists of statements to change the machine's state, i.e. a list of commands.
2) Procedural programing: An imperative paradigm in which sections of the code can be divided into procedures that can be invoked by their names.
3) Functional programming: A software paradigm in which a program is formed out of composing functions without any changes of state and functions are treated as first-class citizens.

(a) The procedural paradigm improves over the imperative programming by allowing the programmer to reduce repetition of code by turning said repetitions into functions. This allows for easier and faster code writing and testing.
(b) Functional paradigm improve the procedural paradigm by making the code have no side effects, therefore any function will behave deterministically, based solely on its input. Allowing for more accurate and efficient unit tests and debugging of the code. Further, this property allows for easy parallelization of code as there are no side effects to reason about. Another benefit is that the function-as-first-citizen style allows for better generalisation, modularization and abstraction of the code.

Question 2

```
type Product = {
    name: string;
    price: number;
    discounted: boolean;
}

//signature: getDiscountedProductAveragePrice(invesntory, number)
// type: (Product[] ) => number
// purpose: calculates the average price of all discounted products in
a given inventory
// pre-conditions: true
// tests: treeToSentence([{name:"a", price:5, discounted:true},
{name:"b", price:10, discounted:true}, {name:"b", price:15,
discounted:true}]) ==> 10
const getDiscountedProductAveragePrice = (inventory: Product[]): number
=> {
    const filteredItems = inventory.filter((p : Product) =>
p.discounted);
    const sum =  filteredItems.reduce((acc, curr) => acc + curr.price,
0);
    const amount =  filteredItems.reduce((acc, curr) => acc += 1, 0);

    return sum / amount;
}
;
```

Question 3

a. [T](x: T[], y: T): boolean {}
b. (x: number[]): number {}
c. [T](x: boolean, y:T[]): T {}
d. [T,S] (f: ((x:number):T):S, g: (x:number):T): (x:number):S {}