1. (a)

i. Imperative: The control flow is an explicit sequence of commands.

ii. Procedural: like imperative programming but organized around hierarchies of nested procedure calls.

iii. Functional: computation proceeds by nested function calls that avoid any global state mutation and through the definition of function composition.

(b) The procedural paradigm improves over the imperative paradigm by adding layers of abstraction in the form of procedures. Procedures interact through well-defined contracts and can encapsulate local variables.

(c) the functional prardigm improves over the procedural paradigm by discouraging the use of shared state and mutation, which makes testing, formal verification and concurrency easier.

2. const getDiscountedproductAverageprice = (inventory : product[])
: number => {

discountedProducts = inventory.filter (product => product.discounted);

discounted prices = discounted products. map (product => product.price);

discountedpricesum = discounted prices. reduce ((sum, price) => sum + price, 0);

return discountedproducts.length == 0 ? 0 :

discountedpricesum / discountedproducts. length ;

};

(a)

3. <T> (x: T[], y: (z: T) => boolean) => boolean

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

(c) <T> (x: boolean, y: T[]) => T

(d) <T1, T2> (f: (b: T1) => T2, g: (a: number) => T1) => (x: number)

=> T2.