**Part 1**

(a) Imperative= a paradigm that explicit with sequence of commands

(b) Procedural=Imperative programming organized around hierarchies of nested procedure calls.

(c) Functional= Computation proceeds by (nested) function calls that avoid any global state mutation and through the definition of function composition.

- Procedural programming is a subset of imperative programming which utilizes subroutines. Also the procedural paradigm is not repeat code like the Imperative paradigm.

-The logic behind the functional paradigm is much clearer and easier to read and understand than the one behind the procedural paradigm.
Moreover, code validation is easier in the functional paradigm than the one in the procedural paradigm.

Q2

```
Function Over60 (grades){

const over60 = (x) => x >60 ;

return filter(over60,grades);
}

Function ReduceOver60(grades){
const sumOfOver60 = reduce(+,0, over60(grades));
return sumOfOver60; }


function Avg(grades){

const avg= ReduceOver60/Over60(grades).length;

return avg;

}
```

Q3

1.Functional definition with sub expression of compound expression.

2. Functional definition with sub expression of functional definition with sub expression of compound expression.

3. Functional definition with sub expression of compound expression of type ternary conditional expression.

4. Functional definition with sub expression of another Functional definition with sub expression of functional invocation with sub expression of functional invocation with sub expression of compound expression.

Q4

An abstraction barrier divides the world into two parts: clients and developers.
The clients don't know how the software works, they just "believe" it does work.
However, the developers do need to know how the software works, and they are the ones who are in charge of the code in the software and make sure it works.