<u>Q1:</u>

Imperative: Control flow is an explicit sequence of commands.

Procedural: Imperative programming organized around hierarchies of nested procedure calls.

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

How does the procedural paradigm improve over the imperative paradigm? How does the functional paradigm improve over the procedural paradigm?

The procedural paradigm improves over the imperative paradigm by:

1) adding layers of abstraction in the form of procedures.
2) procedures interact through well-defined contracts.
3) can encapsulate local variables.

The functional paradigm improves over the procedural paradigm by discouraging the use of shared state that help to do

1) testing
2) the concurrency be easier
3) formal verification.

<u>Q2:</u>

The code :

function averageGradesOver60(grades: number[]) {

```
grades.filter(x=>x>60).reduce((curr,elem)=>curr+elem,0)/grades.filter(x=>x>60)
.reduce((curr,elem)=>curr+1,0)

}
```

<u>Q3:</u>

a) (x, y) => x.some(y) →<T>(x:T[],y:(u:T)=>boolean)=>Boolean)
b) x => x.reduce((acc, cur) => acc + cur, 0) → x:number[] => number
c) (x, y) => x ? y[0] : y[1] → <T>(x:Boolean,y:T[])=>T
d) (f,g) => x => f(g(x+1)) → (f(z:any)=>any,g(l:number)=>any):any=>(x:number):any=>f(g(x+1)):any

<u>Q4:</u>

Explain the concept of “abstraction barriers”?

Answer: abstraction barriers isolate different levels of the system, and the implementer of the high -level system need not know about low-level details.