Q.1:

(a) **Imperative paradigm**: A programming paradigm, in which programs consist of a sequence of instructions, and instructions get executed one at a time, in a set order.

(b) **Procedural paradigm:** A programming paradigm, in which pieces of code can be combined into procedures, which can be called from other places in a program. Such procedures can accept arguments that influence their execution, and return values at the end of it.

(c) **Functional paradigm:** A programming paradigm, in which code is organized into a series of expressions, and makes use of pure functions, that don't have any "side-effects" (no influence on the environment in which they are executed.)

- How does the procedural paradigm improve over the imperative paradigm?
  - Allows code-reuse, structuring the program better visually/logically
  - Allows separation of program's logic from data, allowing for easy changes to the program's beginning state and execution flow
- How does the functional paradigm improve over the procedural paradigm?
  - Allows for safer implementation of parallelism
  - Allows for easier code-testing phase
  - Allows for certain optimizations to be made to functions, knowing that those won't mess up shared memory

Q.2:

```
function averageGradesOver60(grades: number[]) : number {
    const gradesOver60 = grades.filter((num: number) => num > 60);
    return gradesOver60.reduce((sum, curr) => curr + sum, 0)/gradesOver60.length;
}
```

Q.3:

(a)

```
<T1> (x: T1[], y: (elem:T1)=>boolean) => x.some(y);
```

(b)

```
(x: number[]) : number => x.reduce((acc: number, cur: number) => acc+cur, 0);
```

(c)

```
<T1> (x : boolean, y : T1[]) : T1 => x ? y[0] : y[1];
```

(d)

```
<T1> (f: (gRes: T1)=>number, g: (futureArg:number)=>T1) : ((x:number)=>number) =>
x => f(g(x+1));
```

Q.4:

"Abstraction barriers" is a concept, according to which the program logic is abstracted using a certain hierarchy in the program's structure (functions' definition, or data structures). For example using higher-level functions to convey an idea, while those encapsulate in them the usage of lower-level functions to get the job done.