Part 1: Theoretical Questions

Imperative- a programming paradigm that uses statements that change a program's state. an imperative program is a sequence of commands for the computer to perform.

Procedural- a programming paradigm using procedures which contains a series of computational steps to be carried out. A procedure may be called at any point during a program's execution.

Functional- a programming paradigm where programs are constructed by applying and composing functions. A functional program contains a series of expressions to be calculated by the computer.

Procedural improve over Imperative: imperative programming means that the computer get a list of commands and executes them in order, when procedural programming (which is also imperative) allows splitting those instructions into procedures (or functions). That way code may be shortened, and easier to read.

Functional improve over procedural: The advantage of the functional approach is that you have highly re-usable ,and manageable functions which have little to no extra dependencies. This makes your code very easy to maintain and understand. the functional programming paradigm has no side effects, unlike the procedural where we usually have side effects, that way the program could be more parallel than the procedural. Another advantage is type checking (need to check if its type checking or code correctness).

```
<T> (x: T[] , y:(a: T)=>Boolean) => Boolean= (x,y) => x.some(y)
<T> x:T[] =>T =x => x.reduce((acc: T, cur:T) => acc + cur, 0)
<T>(x:boolean, y:T[]) =>T =(x,y)=> x ? y[0] : y[1]
```

Abstraction barriers: Abstraction is the removal of unnecessary detail. The idea is that one person doesn't have to know how any operation does its background work, but really needs to know only what it does. That way, a complex code could get a name, like naming a function, and with a little description any other programmer would know all the necessary details to use that function for his needs. For example, therefore we don't use '0' and '1' in our code, there's a need for separation between an implementor and a client, and as programmers we are clients trying to implement something else.