PPL Home Work 1

Guterman Gregory, Shoham Bracha

March 2023

1.

A. Explain the following programming paradigms:

- Imperative

  Imperative programming is a paradigm that focuses on describing a series of steps to be taken in order to achieve a desired result.
  It involves specifying a sequence of instructions, which are executed in a particular order to manipulate program state, with an emphasis on how to do something.
  Imperative programming languages rely on statements that change a program's state, such as assignments, loops, and conditionals.
  There are no functions or procedures in this paradigm.

- Procedural

  Procedural programming is a paradigm that emphasizes the use of procedures or subroutines, also known as functions, to organize program functionality into reusable code blocks.
  In procedural programming, programs are composed of one or more procedures or routines, which are executed in a specific order to solve a problem.
  There can be side effects in this paradigm.

- Functional

  Functional programming is a paradigm that focuses on the use of functions to solve problems.
  It emphasizes the use of immutable data structures and avoids the use of side effects, which means that functions should not modify the state of the program or the system.
  Functional programming languages treat functions as first-class values, which can be passed as arguments to other functions, returned as values, and stored in data structures.
  This paradigm doesn't allow any side effects.

B. How does the procedural paradigm improve over the imperative paradigm ?

The procedural paradigm improves over the imperative paradigm by organizing the code into reusable units, called procedures or functions.
By breaking down a program into smaller, more manageable parts, procedural programming enables developers to write code that is easier to understand, modify, and maintain.
In addition, procedural programming can help improve the efficiency and speed of a program by reducing the amount of duplicate code and allowing for more efficient memory usage.

C. How does the functional paradigm improve over the procedural paradigm ?

The functional paradigm improves over the procedural paradigm by providing a more declarative approach to programming.
Rather than specifying how to solve a problem, functional programming focuses on what needs to be done, and leaves the details of how to do it to the underlying language and libraries.
This approach can help reduce the complexity of a program by making it easier to reason about and understand.
Additionally, functional programming emphasizes immutability and the avoidance of side effects, which can lead to more reliable and predictable programs.
Finally, functional programming languages often provide powerful abstractions, such as higher-order functions, closures, and monads, which can make it easier to express complex ideas and solutions in a concise and elegant way.

2. Convert the following function to adhere to the Functional Programming paradigm, using some or all the functions we saw in class: map, filter, reduce:

```
reduce(filter(map(numberAsString, parseInt), (x: number) => x % 2 === 0 ), (acc, x) => acc + x), 0)
```

3. Write the most specific types for the following expressions:

```
(a) <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
```

4. Explain the concept of “abstraction barriers”:

It is the concept main idea whom is hiding all but the relevant data about an object to reduce complexity and increase efficiency, Abstraction barriers are used to divide a system (program) into levels each with a well-defined interface that isolates it from the complexities of other components. They also make it easier to modify or replace individual components without affecting the rest of the system.
By using abstraction barriers, software designers can create systems that are easier to understand, modify, and maintain. They also make it easier to test individual components and ensure that they work correctly.