Tecnológico Nacional de México
Instituto Tecnológico de Tijuana

Subdirección Académica
Departamento de Sistemas y Computación

Semestre:
Agosto – Diciembre 2021

Carrera:
Ingeniería en Tecnologías de la Información y Comunicaciones
Ingeniería en Sistemas Computacionales

Materia y serie:
Datos Masivos BDD-1704TI9A

Unidad a evaluar: Unidad I

Nombre de la Tarea:
Práctica 4

Nombre del Alumno:
Sifuentes Martinez Manuel Javier 17212934
Flores Gonzalez Luis Diego C16211486

Nombre del docente:
José Christian Romero Hernández

Practice 4

Given the pseudocode of the Fibonacci sequence in the link provided, implement with Scala Algorithm 1, Algorithm 2.

Algorithm 1:
This type of algorithm generates a large number of sums, so it is only recommended for small numbers of n. Within the function, it is observed that recursion is used when calling the same function, but modifying the value of n.

```
Algorithm 1:
def fib1(n: Int): Int = {
    if(n < 2) return n else fib1 (n-1) + fib1 (n-2)
}

scala> fib1 (15)
res22:carry Int = 610
```

Algorithm 2:
Toout this algorithm additional libraries were needed for use of the square root and the empowerment of the numbers, once imported, the complexity only lies in the operations that are inside the else clause, assigning values to variables and using multiple operations to reach the objective that is to return the correct value of Fibonacci.

```
import scala.math.sqrt
import scala.math.pow

def fib2(n: Int): Int = {
    if(n < 2) {
        return n
    }
    else {
        var p = ((1+ sqrt (5)) /2)
        var j = ((pow (p, n) - pow (1-p, n)) / sqrt (5))
        return (j) .toInt
    }
}

scala> fib2 (15)
res8: Int = 610
```

Algorithm 3:
In the following algorithm, the initialization of the number to find its number in fibonacci is performed as well as the value of a of 0, b of 1 and c from 0, where with the use of a for it is possible to change the value of the previous variables, where in each iteration the value of a changes until the fibonacci cycle is completed.

```
def fibo_3(n: Int): Int = {
```

```
      var n1 = n - 1;
      var a = 0;
      var b = 1;
      var c = 0;
      for (k <- 0 to n1)
      {
        c = b + a;
        a = b;
        b = c;
      }
      return a
    }
```

Result:

```
scala> fibo_3 (15)
res1: Int = 610
```

Algorithm 4:

The following algorithm shows how the number is obtained in fibonacci where in the cycle the sum of the previous variables is performed to obtain the equivalence for get the correct number in variable a.

```
def fibo_4(n: Int): Int = {
      var n1 = n -1;
      var a = 0;
      var b = 1;
      for (k <- 0 to n1)
      {
        b = b + a;
        a = b - a;
      }

    return(a)
    }
```

Result:

```
scala> fibo_4 (15)
res2: Int = 610
```

Algorithm 5:

In the following algorithm it works with an if and else, where the first one is to verify if the number is less than 2 then returns the evaluated number , for the else, the array containing the number of rows equal to the number to evaluate plus 1 is defined, for the spaces of 0 and 1 the value is previously defined and in the for loop the array is filled with the sequence of the positions Previous of each position minus 1 and minus 2 respectively to return the position of the array according to the number to evaluate at the end.

```
def fibo_5(n: Int): Int = {
      if (n < 2) return n
      else {
        var z = new Array[Int] (n + 2);
        z (0) = 0;
        z (1) = 1;
        for (k <- 2 to (n + 1))
          {
            z (k) = z (k - 1) + z (k - 2);
          }
        return(z (n))
      }
    }
```

Result:

```
scala> fibo_5 (15)
res3: Int = 610
```