IMPERATIVE PROGRAMMING 2017

MOISE GABRIEL

QUESTION 1

```
object Base {
    var base = 7
    type number = Array[int]
    // (a)
    def copyNum (a : number) : number = {
        var b = new number(a.size)
        for (i <- 0 until a.size) b(i) = a(i)
        b
    }
    // (b)
    def lengthen (a : number) : number = {
        var N = a.size
        N = 2 * N
        var res = new number(N)
        for (i <- 0 until a.size) res(i) = a(i)
        res
    }
    // (c) Here, I suppose that normalizing is transforming a "number" in its decimal representation
    def normal (a : number) : int = {
        var N = a.size
        var baseMul = 1
        var value = 0
        for (i <- 0 until N) { value = value + a(i) * baseMul ; baseMul = baseMul * base }
        value
    }
    // (d)
    def makeNum (value : int) : number = {
        if (value == 0) { var b = new number(1) ; b(0) = 0; return b }
        // Finding the number of digits needed for the representation in base "base" of value
        var aux = value
        var i = 0
```

```
    while (aux != 0)  {aux = aux / base ;  i+=1}
    var N= i  //the size of the "number"
    var b= new number(N)
    i = 0 ; aux = value
    while (i < N)
    { b(i) = aux % base
      aux = aux / base
      i += 1
    }
    b
  }
// (e)
def add (a : number, b: number) : number ={
     var result = normal(a) + normal(b)
     return makeNum(result)
}
// (f)
def mul (a: number, b: number): number ={
     var result = normal(a) * normal(b)
     return makeNum(result)
}
// (g)
def outNum(a: number) : String ={
     var result = normal(a)
      return result.toString
}
```