



Universidad Nacional Autónoma de México

Facultad de Ingeniería

Cómputo móvil

Medel Sánchez Berenice

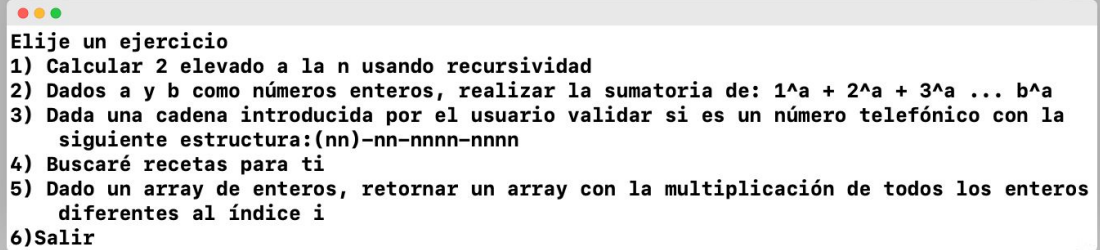


Actividad 2

Para el desarrollo de los ejercicios se creó una “command line application” para tener acceso a la lectura de datos desde consola. Se realizó un menú en el cual se puede seleccionar cada uno de los ejercicios, hasta que el usuario decida salir de la ejecución. Cada ejercicio se encuentra dentro de su propio archivo con extensión swift y los métodos son invocados desde el archivo main.swift creando una instancia de las struct ExerciseOne, ExerciseTwo, etc.

```
repeat {
    print("""
        \n\nElije un ejercicio\n\
        1) Calcular 2 elevado a la n usando recursividad\n\
        2) Dados a y b como números enteros, realizar la sumatoria de: 1^a + 2^a + 3^a ...
        b^a\n\
        3) Dada una cadena introducida por el usuario validar si es un número telefónico
        con la siguiente estructura:\
        (nn)-nn-nnnn-nnnn\n\
        4) Buscaré recetas para ti\n\
        5) Dado un array de enteros, retornar un array con la multiplicación de todos los
        enteros diferentes al índice i\n\
        6) Salir
        """)
    exercise = Int(readLine() ?? "6")!
    switch exercise {
    case 1:
        print("Dame un entero n mayor o igual a 0\n")
        n = Int(readLine()!)!
        if n >= 0 {
            print("2^\(n) = \(ExerciseOne().run(n: n))")
        } else {
            print("valor inválido")
        }
    case 2:
        print("Dame a:")
        a = Int(readLine()!)!
        print("Dame b:")
        b = Int(readLine()!)!
        print("Resultado: \(ExerciseTwo().run(a: a, b: b))")
    case 3:
        phoneNumber = readLine()!.trimmingCharacters(in: .whitespacesAndNewLines)
        print(phoneNumber)
        if ExerciseThree().run(phoneNumber: phoneNumber){
            print("Número válido")
        } else {
            print("Número inválido")
        }
    case 4:
        print("Dame los ingredientes separados por coma y un espacio")
        ingredients = readLine()!
        recipes = ExerciseFour().run(ingredients: ingredients.components(separatedBy:
        ", "))
        if recipes.isEmpty {
            print("No encontré recetas :(")
        } else {
            recipes.forEach({print($0)})
        }
    case 5:
        print("Dame un array de numeros separados por coma")
        numbers = readLine()!.components(separatedBy: ",").map({Int($0)!})
        print("Result: \(ExerciseFive().run(numbers: [1, 2, 3, 4, 5]))")
    case 6:
        print("Adios :)")
    default:
        print("Opción inválida")
    }
} while exercise != 6;
```

Imagen 1. Código del archivo main.swift



```

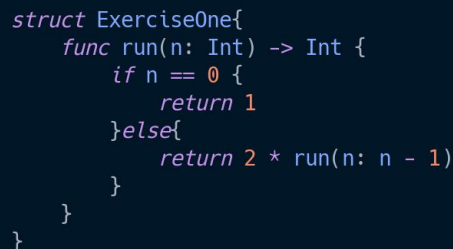
Elige un ejercicio
1) Calcular 2 elevado a la n usando recursividad
2) Dados a y b como números enteros, realizar la sumatoria de: 1^a + 2^a + 3^a ... b^a
3) Dada una cadena introducida por el usuario validar si es un número telefónico con la
   siguiente estructura:(nn)-nn-nnnn-nnnn
4) Buscaré recetas para ti
5) Dado un array de enteros, retornar un array con la multiplicación de todos los enteros
   diferentes al índice i
6) Salir

```

Imagen 2. Funcionamiento del menú general

1. *Desarrolle el algoritmo que calcule 2 a la N donde N es un número entero mayor o igual a cero, el algoritmo debe ser desarrollado utilizando exclusivamente recursividad*

Para este ejercicio se llama de forma recursiva a la función run con un índice $n - 1$, multiplicando este resultado por 2. La condición que finaliza el método run es cuando $n=0$, retornando un 1.

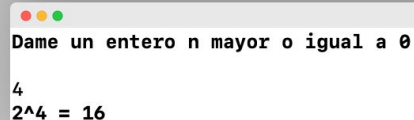


```

struct ExerciseOne{
    func run(n: Int) -> Int {
        if n == 0 {
            return 1
        }else{
            return 2 * run(n: n - 1)
        }
    }
}

```

Imagen 3. Código ExerciseOne.swift



```

Dame un entero n mayor o igual a 0

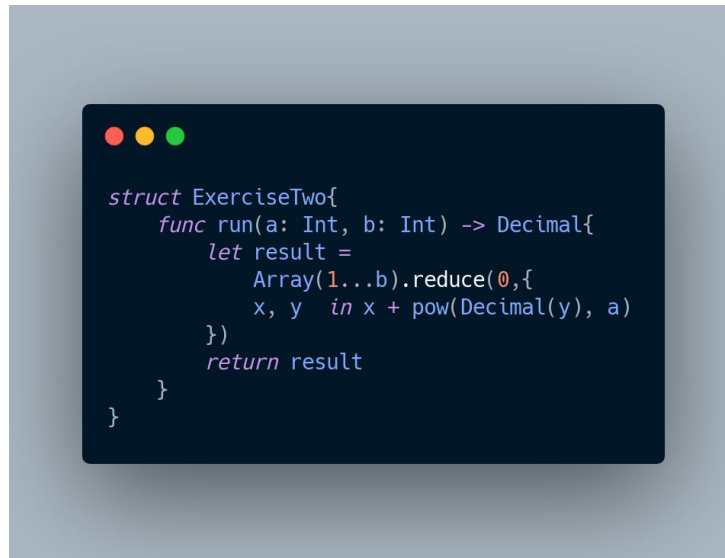
4
2^4 = 16

```

Imagen 4. Ejecución del ejercicio 1

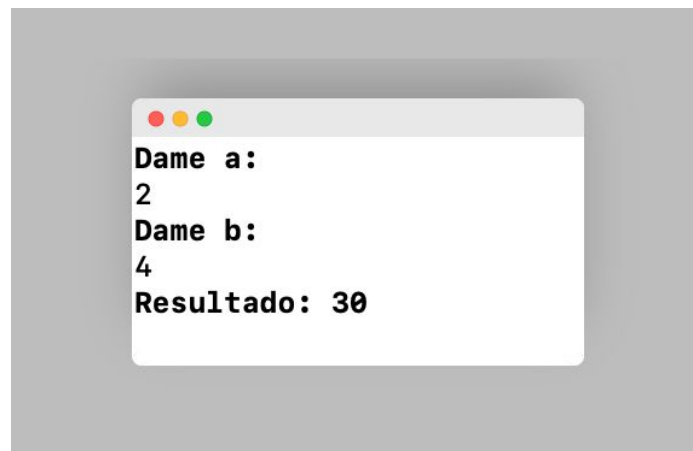
2. Dado a y b como números enteros, realizar la sumatoria de: $1^a + 2^a + 3^a \dots b^a$

En este ejercicio se utilizó un reduce para sumar las potencias desde 1 hasta b elevadas a la base a .

A screenshot of a code editor window with a dark blue background. It contains Swift code for a struct named ExerciseTwo. The code defines a function run(a: Int, b: Int) that returns a Decimal. Inside the function, it calculates the sum of powers from 1 to b using the reduce method on an array. The code is as follows:

```
struct ExerciseTwo{  
    func run(a: Int, b: Int) -> Decimal{  
        let result =  
            Array(1...b).reduce(0,{  
                x, y in x + pow(Decimal(y), a)  
            })  
        return result  
    }  
}
```

Imagen 5. Código ExerciseTwo.swift

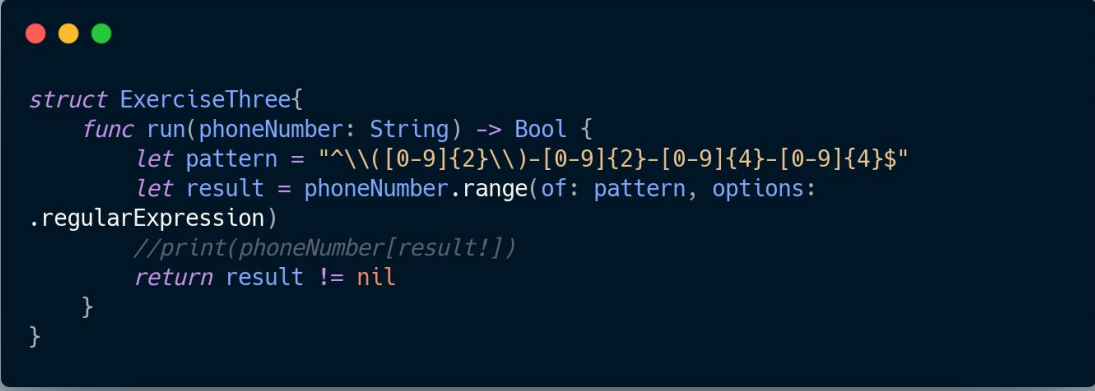
A screenshot of a program execution window with a light gray background. It shows the input values for 'a' and 'b' and the resulting output. The text is as follows:

```
Dame a:  
2  
Dame b:  
4  
Resultado: 30
```

Imagen 6. Ejecución del ejercicio 2

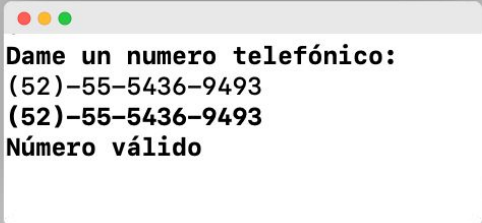
3. Dada una cadena introducida por el usuario validar si es un número telefónico con la siguiente estructura: (nn)-nn-nnnn-nnnn

Para validar la estructura del número telefónico se hizo uso de expresiones regulares, se validó el patrón: `^\([0-9]{2}\)-[0-9]{2}-[0-9]{4}-[0-9]{4}$` mediante el uso del método `range` con la opción `.regularExpression`.



```
struct ExerciseThree{
    func run(phoneNumber: String) -> Bool {
        let pattern = "^\([0-9]{2}\)-[0-9]{2}-[0-9]{4}-[0-9]{4}$"
        let result = phoneNumber.range(of: pattern, options:
        .regularExpression)
        //print(phoneNumber[result!])
        return result != nil
    }
}
```

Imagen 7. Código ExerciseThree.swift



```
Dame un numero telefónico:
(52)-55-5436-9493
(52)-55-5436-9493
Número válido
```

Imagen 8. Ejecución del ejercicio 3

4. Dada una lista de recetas (mínimo 10 recetas - texto) encontrar las recetas que contengan alguno o algunos ingredientes.

En este ejercicio se hizo uso de la función filter, a la cual se le pasó un closure que hacía uso de las funciones contains y la opción where.

```
struct ExerciseFour{
    let recipes: [String] = [
        """
        Pay de Limon \n\
        Ingredientes\n\
        Batido de limón:\
        180 g Queso Crema Philadelphia\
        1 lata de leche condensada\
        1 lata de leche evaporada\
        ½ tza. de jugo de limón\
        1 cdta. de ralladura de limón para decorar\
        2 tzas. de galleta molida\
        90 g de mantequilla
        """
    ]
    func run(ingredients: [String]) -> [String] {
        let lowercased_ingredients = ingredients.map({$.lowercased()})
        let result = recipes.filter({lowercased_ingredients.contains(where:
        $.lowercased().contains)})
        return result
    }
}
```

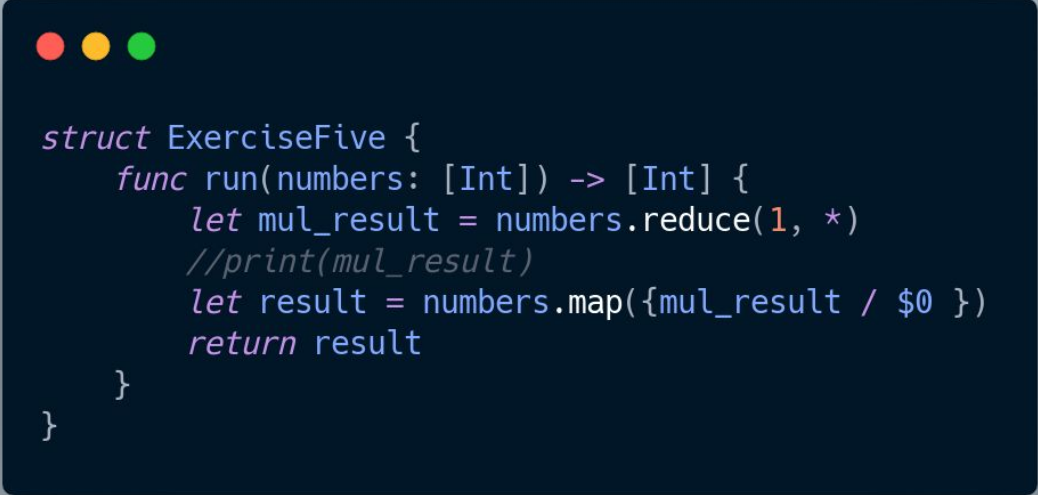
Imagen 9. Código ExerciseFour.swift

```
Dame los ingredientes separados por coma
naranja,tequila
SUBMARINO - Ingredientes: Tequila 30ml, Cerveza oscura 180ml - Procedimiento: Rellenar el
caballito con tequila, colocarlo dentro del vaso old fashioned y rellenar con cerveza
oscura.
SUNRISE - Ingredientes: Tequila reposado 60ml, jugo de naranja 100ml, granadina 30ml, hielo
100g - Procedimiento: Agregar en el vaso Collins hielo, el tequila y rellenar con jugo
de naranja y mandar al fondo la granadina con ayuda de una bailarina
BRASS MONKEY - Ingredientes: Ron oscuro 32ml, zumo de naranja 60ml, jarabe natural 15ml,
rodaja de naranja 5g, cerveza oscura 150ml - Procedimiento: Agregar ron oscuro, zumo de
naranja y jarabe natural y rellenar con cerveza oscura. Decorar con rodaja de naranja.
```

Imagen 10. Ejecución del ejercicio 4

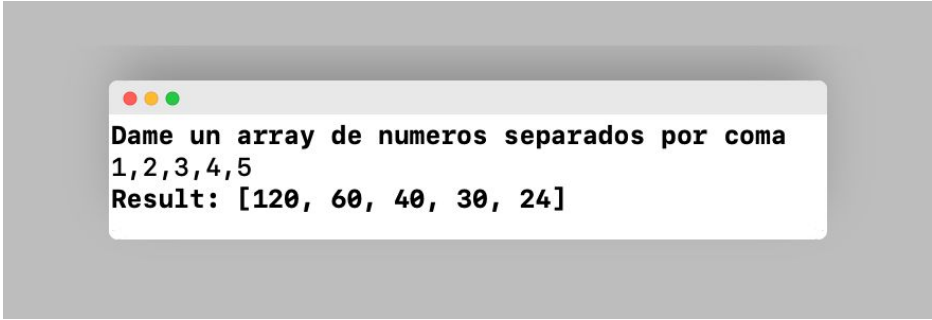
5. *Dado un arreglo de números enteros, regresar un nuevo arreglo tal que cada elemento en la posición "i" del nuevo arreglo es el producto de todos los números del arreglo original menos el de la posición i.*

En este ejercicio se utilizó un reduce para obtener el total de la multiplicación de todos los elementos del array original. Posteriormente se utilizó un map iterando sobre el array original y dividiendo el total entre el contenido i del array, obteniendo así la multiplicación de todos los números del array excepto el del índice i.



```
struct ExerciseFive {  
    func run(numbers: [Int]) -> [Int] {  
        let mul_result = numbers.reduce(1, *)  
        //print(mul_result)  
        let result = numbers.map({mul_result / $0 })  
        return result  
    }  
}
```

Imagen 11. Código ExerciseFive.swift



```
Dame un array de numeros separados por coma  
1,2,3,4,5  
Result: [120, 60, 40, 30, 24]
```

Imagen 12. Ejecución del ejercicio 5