UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL RESISTENCIA

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

PARADIGMAS DE PROGRAMACIÓN
Guía de Trabajos Prácticos

Equipo docente:

Director de Cátedra

Nombre y Apellido: Dr. Rubén Alfredo Bernal

Categoría docente: Prof. Asociado

e-mail: rbernal@frre.utn.edu.ar; rbernal73@gmail.com

Docentes por División

División "A"

Teoría: Dr. Rubén Alfredo Bernal — e-mail: rbernal@frre.utn.edu.ar

Práctico: Ing. Carlos Cuevas — e-mail: cac@frre.utn.edu.ar

División "B"

Teoría: Dr. Rubén Alfredo Bernal — e-mail: rbernal@frre.utn.edu.ar

Práctico: Ing. Cesar Javier Pizzi — e-mail: cesarjpizzi@yahoo.com.ar

CICLO LECTIVO 2022

Trabajo Práctico 1	*CONCEPTOS DE SMALLTALK*

1. Evalúe las siguientes expresiones en el entorno y anote los resultados, entendiendo en cada caso su funcionamiento.

    a. 5+6
    b. 20 factorial
    c. 'Esto es una prueba' size
    d. #(1 3 5 7) at: 2
    e. ‘Paradigmas’ isArray
    f. 5 * 7
    g. 5 // 2
    h. 4 \\ 3
    i. 2 / 6
    j. 1.5 + 6.3e2
    k. Array new
    l. Date today
    m. Time now

2. Evalúe los siguientes Mensajes Unarios.

    a. #('arreglo' 'de' 'strings') size
    b. 'Hoy es Jueves' asUpperCase
    c. 'hola aquí estoy' reversed
    d. #(4 'cinco' 6 7) reversed
    e. $A asciiValue
    f. 65 asCharacter
    g. ‘cuál es la longitud de esta oración?’ size

3. Evalúe los siguientes Mensajes Binarios.

    a. 'hola', ' aquí estoy'
    b. #(1 2 3), #(4 5 6)
    c. 4 = 5
    d. #(1 2 $a), #($b ‘cd’)

4. Evalúe los siguientes Mensajes de Palabra Clave.

    a. 'Esto es una prueba' at: 3
    b. 'Hola' includes: $o
    c. 'hola' at: 1 put: $H
    d. 'Paradigmas de Programación' copyFrom: 4 to: 9
    e. #(9 8 7 6 5) at: 3
    f. #(1 (2 3) 4 5) includes: #(2 3)
    g. #(9 8 7 6 5) copyFrom: 1 to: 2
    h. Array new: 10

5. Mensajes anidados. Evaluarlos y de ser necesario, colocar paréntesis para mostrar de que manera ST evalúa las expresiones.

    a. 'hola' size + 4
    b. 'ahora' size + #(1 2 3 4) size
    c. #(1 12 24 36) includes: 4 factorial
    d. 3 + 4 * 2
    e. 3 + (4 * 2)
    f. 4 factorial between: 3 + 4 and: 'hola' size * 7
    g. 'hola' at: (#(5 3 1) at: 2)
    h. 6 + 9 asString
    i. Array with: 1 with: 'hola' with: (1/3)

6. Variables Globales, siempre empiezan con mayúscula. Evaluarlas una a una verificando los resultados.

    a. Display
    b. Transcript
    c. Debug
    d. Disk
    e. Distancia
    f. Distancia:= 15

7. Comparando Objetos. Evaluar c/u, verificar resultados y en caso de error indicar la expresión correcta.

    a. 3 < 4
    b. #(1 2 3 4) = #(1 2 3 4)
    c. 'hola' <= 'adios'
    d. 5 = 2 + 3
    e. 5 = (2 + 3)

8. Bloques de código. (Recordar que un bloque puede tener argumentos)

    a. [$a isVowel] value
    b. [$b isVowel] value
    c. [3+4 . 'hola' asUpperCase] value
    d. | bloque |
    bloque:=['Hola ', ' como estás ?'].
    ^bloque value.
    e. [:c | c isVowel] value: $a
    f. [:c | c isVowel] value: $b
    g. | bloque resp |
    bloque:=[:a :b | a , b].
    resp:=bloque value: 'Hola ' value: ' como estás ?'.
    ^resp

9. Expresiones Booleanas

    a. 5 < 2 or: [$a isVowel]
    b. 5 < 2
    c. (5 < 2) not
    d. 5 < 2 and: [$a isVowel]
    e. (5 < 2) not and: [$a isVowel]
    f. (5 < 2) not or: ['hola' size < 2 and: [$a isVowel]]

10. ¿Cómo sabemos a qué clase pertenece cada objeto? Evalúa las expresiones:

    a. #(Francesca Jackie Marisa Bree) class
    b. 'Rakesh Vijay Charles Daniel Tyler' class
    c. 5 class
    d. (1/2) class
    e. 5.2 class

11. Mensajes en cascada, llamamos así al conjunto de mensajes que enviamos a un mismo objeto. Verifica estas dos expresiones y explica la diferencia.

    3 factorial; factorial; factorial
    3 factorial factorial factorial

12. ST tiene una ventana llamada "Inspector" que permite ver y cambiar las variables de instancia de un objeto. Evalúa este código y di que observas.

```
| a |
a := #( 1 2 sam 'joe' ( 4 5 ) ).
a at: 2 put: 3 / 4.
```

a inspect

13. Condicionales. Verifica estas expresiones

a. 3 < 4 ifTrue: ['el bloque verdadero']
ifFalse: ['el bloque falso']

b. (5>2) ifTrue:[^'5 es MAYOR que 2']
ifFalse:[^'5 es menor que 2'].

c. $b isVowel ifTrue:[^'es una vocal']
ifFalse:[^'es una consonante']

Trabajo Práctico 2 — *SMALLTALK. EXPRESIONES MATEMÁTICAS.*

1. Crear variables temporales y referencias de estas a distintos objetos (enteros, reales, caracteres, flotantes, cadena de caracteres, etc.)
2. Realizar las operaciones F/G y F*G. Utilizando sumas y restas sucesivas.
3. Realizar x^y.
4. Solicitar al usuario que ingrese los coeficientes de una ecuación cuadrática, calcular sus raíces y mostrarlas.
5. Solicitar el ingreso de un nº y verificar si este es o no primo.
6. Idem anterior, decir si es par o impar.
7. Dado un número determinar sus múltiplos.
8. Escribir un programa que ingrese un listado de números e informe la cantidad de múltiplos de 2, 3, 5 y 7.
9. Un número entero positivo se dice perfecto si es igual a la suma de todos sus divisores, excepto el mismo. Ejemplo: los números 6 (1+2+3), 28 (1+2+4+7+14) y 496 (1+2+4+8+16+31+62+124 +248) son perfectos. Escriba un método booleano que permita diferenciar si un número (único parámetro) es perfecto.
10. Dos números se dicen amigos cuando uno de ellos es igual a la suma de todos los divisores del otro excepto el mismo. Ejemplo: los números 220 (1+2+4+5+10+11+20+22+44+55+110=284) y 284 (1+2+4+71+142=220) son amigos. Escriba un método booleano que permita discernir si dos números (parámetros) son amigos.
11. Ingresar 2 polinomios y realizar la suma y el producto de ambos.
12. Se tiene un arreglo de n números naturales que se quiere ordenar por frecuencia, y en caso de igual frecuencia, por su valor. Por ejemplo, a partir del arreglo [1, 3, 1, 7, 2, 7, 1, 7, 3] se quiere obtener [1, 1, 1, 7, 7, 7, 3, 3, 2].
13. Realizar un algoritmo que lea una serie de números reales y verifique si están ordenados ascendentemente o no, informando por pantalla.

Trabajo Práctico 3	*SMALLTALK. EXPRESIONES CON CADENAS DE CARACTERES*

1. Convertir una cadena a mayúsculas y minúsculas.
2. Dada una cadena de entrada, devolver otra en la que los caracteres en mayúsculas hayan sido cambiados por caracteres en minúsculas y viceversa.
3. Verificar si una frase es un palíndromo o no.
4. Contar la cantidad de vocales de una frase.
5. Ingresar dos cadenas y devolver una 3º que contenga los elementos de las 2 anteriores pero intercalados.
6. Dada una cadena de entrada, devolver otra en la cual las palabras estén en formato 'tipo título'.
7. Convertir un número en el sistema decimal al sistema binario.
8. Dada una frase contar la cantidad de palabras en mayúsculas.
9. A partir de una frase ingresada por el usuario contar la cantidad de palabras que empiezan con una determinada letra (también ingresada por el usuario).
10. Ídem anterior, pero contar la cantidad de palabras que terminan con una letra determinada.
    a. Dado un texto, se pide:
        i. la posición inicial de la palabra más larga,
        ii. la longitud del texto,
        iii. cuantas palabras con una longitud entre 8 y 16 caracteres poseen más de tres veces la vocal 'a'
11. Escribir un subprograma que dado n, lea n caracteres que forman un número romano y que devuelva un string que represente a dicho número romano y un número que represente el equivalente decimal.
12. Dado un texto terminado en punto, determinar cuál es la vocal que aparece con mayor frecuencia.
13. Dado un texto terminado en '/' se pide determinar cuántas veces aparece determinada letra, leída de teclado.
14. Dado un texto terminado en '/' determinar cuántas veces tres palabras seguidas comienzan con la misma letra.
15. Leer dos letras de teclado y luego un texto terminado en '/'. Se pide determinar la cantidad de veces que la primera letra precede a la segunda en el texto.

Trabajo Práctico 4 — *SMALLTALK. CLASES Y MÉTODOS*

1. Existen clases definidas en ST que permiten visualizar distintos tipos de ventanas solicitando información al usuario o informando al mismo. Evalúa estas expresiones.

```
1. UIManager default request: '¿Cuál es tu edad ? '
2. |nombre|
   nombre:=UIManager default request:'¿Cómo te llamas? '.
   ^UIManager inform: 'Hola ', (nombre asString).
c. |resp|
   resp:= UIManager confirm:'Hoy es tu cumpleaños??'.
   resp ifTrue:[UIManager inform:'FELIZ CUMPLE!!']
```

2. Métodos Iterativos o estructuras de control. Recordar la diferencia entre cada uno de ellos y verificar cada trozo de código, en caso de error corregirlo y explicar que sucede. (Prestar atención a las variables de bloque)

a.
```
| suma  cont|
suma:=0.
cont:=1.
[cont<11]whileTrue:[suma:=suma+cont.
                                cont:=cont+1].
^suma
```

b.
```
|a|
a:=0.
6 timesRepeat:[a:=a+2].
^a
```

c.
```
|x|
x:=0.
1 to:5 do:[:i| x:=x+i].
^x
```

d.
```
| cad|
cad:='Paradigmas de Programación'.
1 to:(cad size) by:2 do:[:i| cad at:i put:$-].
^cad
```

e.
```
| x|
x:='Paradigmas de Programación'.
(x size) timesRepeat:[:i| x at:i put:$-].
^x
```

3. Escribir una función que permita, dado un número de tres dígitos escrito en palabra, convertirlo en su valor numérico del tipo entero. Ej.: la entrada doscientos veinticinco producirá la salida 225.
4. En SmallTalk existe un método llamado substrings, defina su propio método que realice exactamente la misma tarea.
5. Pasar un período expresado en segundos a un período expresado en días, horas, minutos y segundos.

6. Hacer un método que dada una fecha en formato DD/MM/AA, verifique si es correcta o errónea. Ej.: El 31/02/97 es una fecha errónea.
7. Leer datos de lados de un triángulo A, B, C; haciendo consistencia de datos (Debe ser A < B + C si A es el lado mayor). Informar según sea A^2 con respecto a B^2 + C^2, si el triángulo es rectángulo, acutángulo u obtusángulo.
8. Desarrollar un método que devuelva en un vector los números primos entre 2 y 200.
9. Implemente una clase para representar pilas con las operaciones apilar, desapilar, tope, vacia.
10. Construir un método 'longitudes' tal que, dada una lista de cadenas, devuelva una lista de pares ordenados que representen la ubicación de cada cadena y su longitud. Ej.: longitudes ['este', 'es', 'el', 'ejemplo', 'sí?']. Devolverá: [(1,4),(2,2),(3,2),(4,7),(5,3)]
11. Construir un método llamado 'masLarga', tal que este reciba una lista de cadenas y retorne aquella que sea l la más larga de todas. Ej.: masLarga:['este', 'es', 'el', 'ejemplo', 'sí?']. Devolverá: 'ejemplo'.
12. Definir un método tal que, dado un texto y una lista de caracteres, retorne el texto depurado, sin las repeticiones consecutivas de los caracteres dados.

    Ej.: depura 'hola,,, todo bien,,, y uds..?????' [',' ','.','?'] Devuelve: 'hola, todo bien, y uds.?'
13. Realizar un método que recibiendo una cadena de palabras y un valor entero n, permita devolver las n palabras mayores en forma de tupla junto con su posición y su longitud.

    Ej.: metodoX ['mariano', 'mariposa','trebol','aire'] 2.

    Resultado: [('mariposa',2,8),('mariano',1,7)]
14. Hacer un método que dado un conjunto de palabras, permita armar otra; esta se formará con las iniciales de las palabras que empiezan con una consonante y con la última consonante.

    Ej.: MetodoY ['universidad', 'tecnológica', 'nacional']. Devolverá: 'DTNL'
15. Definir todos los métodos que considere necesario, tal que dada una lista de cadenas, y otra lista de enteros, devuelva la lista de aquellas cadenas cuya longitud coincide con el entero correspondiente en la segunda lista.

    Ej.: consistentes ['yo', 'no','te', 'digo', 'que', 'no'] [2,5,3,7] . → ['yo']
16. Definir un método cuentaChar, tal que dada una lista de strings y un carácter, devuelva una lista compuesta por listas de dos elementos; cada una de ellas conteniendo el string original y la cantidad de veces que aparece el carácter en ella.

    Ej.: cuentaChar ["Marta","Lucía", "Rodrigo"] 'r' .→ [["Marta","1"],["Lucía","0"],["Rodrigo","1"]]
17. Se necesita un método llamado "ocurrencias", tal que dada una palabra, devuelva una lista de pares ordenados [(caracter,repeticionesDelCaracter)] .

    Ej.: ocurrencias "lirio" . Retornará: [('l',1), ('i',2), ('r',1), ('o',1)]
18. Hacer un método que reciba una cadena de enteros y un número entero, y devuelva la cantidad de múltiplos de ese número en la lista. Ej.: MétodoXY [2,8,5,9,10,11,20] 5 → 3 .
19. Idem anterior, pero que retorne la lista de los múltiplos.
20. Definir un método tal que, dada una lista de cadenas, devuelva la misma lista con las cadenas ordenadas según su longitud. Si hay dos cadenas de igual longitud, es indistinto cuál queda primera.

    Ej.: ordena ["hola", "como","estas","hoy"] → ["estas","hola","como","hoy"]
21. Dividir una cadena en n cadenas de m caracteres. Lo restante se elimina.

    Ej.: métodoZ "abcdefg" 2 3→ ["abc","def"] ; métodoZ "abcde" 3 1 → ["a", "b", "c"]
22. Dada una lista de enteros, devolver una lista donde el primer elemento sea la suma en absoluto de los negativos. El resto de la lista se compone por los elementos positivos de la lista original.

    Ej: metodoX [-1, 2, 10, -4] → [5,2,10]; metodoX [3, -2, 6, 4, -2] →[4, 3, 6, 4]
23. Escribir un método sacarVocales, tal que dada una cadena de palabras devuelva una cadena con las mismas palabras, pero solo las consonantes.

    Ej.: sacarVocales ["Luis" , "Antonio" , "Marta"] → ["Ls" , "ntn" , "Mrt"]

24. Escribir un programa que cargue en un vector llamado bisiestos todos los años bisiestos correspondientes al intervalo de tiempo que el usuario elija. ¡Recordar!: Un año es bisiesto cuando es múltiplo de cuatro, exceptuando los múltiplos de 100 que no lo sean de 400. Ej.: son bisiestos: 1988, 1992, 2000 pero estos no lo son: 1800, 1853, 1900.
25. Construya un método cargaMatriz, de tal manera que el usuario indique cantidad de columnas y de filas, y luego ingrese los elementos de la misma. Verifique al final, que se pueda acceder a los elementos de las matriz a través del método at:at: ya definido en ST.
26. Solicite al usuario el ingreso de dos matrices de la misma dimensión m x n y realice la suma entre ambas.
27. Idem anterior, pero realizar el producto entre las matrices.

Ejercicios Complementarios

28. Existe una empresa de comunicaciones, que necesita tarifar las llamadas de sus clientes. La misma posee una serie de antenas, dispuestas de tal manera, que existe una cobertura en los emplazamientos que el cliente se mueve (provincial/nacional).

    Cada cliente posee un dispositivo, el cual tiene una posición de antena origen, por otra parte, este dispositivo está en todo momento bajo la cobertura de una antena específica. Tanto antenas como dispositivo se referencian por coordenadas (x,y).

    Se debe realizar la jerarquía de clases, el desarrollo de las clases, y los siguientes métodos obligatorios: actualizarPosicion, agregarAntena, eliminarAntena, antenaDeReferencia, calcularComunicación, mostrarItinerario.

29. Considere la gestión de ventas de pasajes de larga distancia, donde se tienen que asentar datos del viaje y realizar comprobaciones de alguna parte de ellos.

    Datos del Servicio: fecha, hora, línea, origen, destino, asiento, tipo de servicio, precio. El tipo de servicio cambia continuamente y el mismo contiene la información de los recursos asignados en cada servicio: coche, chofer, películas, etc. Existen de tipo normal, executivo, premium.

    Datos del Pasajero: DNI, nombre, domicilio.

    Validaciones: línea, comprobar horarios, disponibilidad de asientos.

    Elaborar el diagrama de clases, indicar herencias, colaboraciones y cualquier otro recurso utilizado.

    Implementar la consulta de un servicio, el alta, desactivación y borrado.

    Generar la abstracción de los tipos de servicios.

Trabajo Práctico 5	*PROGRAMACIÓN FUNCIONAL*

1. Problemas aritméticos.

1.1. Definir recursivamente el producto en función de la suma.

1.2. Escribir una función **cociente** que calcule el cociente de una división entera utilizando sumas y restas.

1.3. Escribir una función **resto** que calcule el resto de una división entera utilizando restas únicamente.

1.4. Escribir una función **fact** que calcule el factorial de un número.

1.5. Definir recursivamente:

$$\sum_{i=1}^{n} i$$

1.6. Definir recursivamente la potencia en función del producto.

1.7. Definir recursivamente el combinatorio de m tomados de a n:

$$C_n^m = \frac{n!}{(n-m)!m!}$$

1.8. Definir recursivamente:

$$\sum_{i=1}^{n} (3*i)+1 \qquad \text{y} \qquad \sum_{i=1}^{n} (3*i)-1$$

1.9. Escribir una función **binario** que calcule la representación binaria de un número entero dado en representación decimal.

1.10. Escribir una función tal que, dado un par de enteros, devuelva el valor de verdad de la relación de mayor estricto, utilizando solo comparaciones con el cero. Definir rango y dominio de la función.

1.11. Escribir un programa que devuelva el enésimo término de la serie de Fibonacci.

1.12. Escribir una función Haskell que determine si un número p es primo.
Nota: un número natural p es primo si es mayor que 1 y no existe otro número natural n > 2 y `n^2 <= p` sea divisor de p.

2. Problemas con listas

2.1. Dada la siguiente función

```
fi [xs] y = []
fi (x:xs) y = x : fi xs y
```

Decir cuales de las siguientes afirmaciones son verdaderas:

2.1.a. fi "abcd" 'a' R = "ab"
2.1.b. fi "abcd" 'e' R = "abcde"
2.1.c. fi "edcb" 'a' R = "edcb":'a'
2.1.d. fi "abcd" 'a' R = "aba"
2.1.e. fi "abcde" 'f' R = "abc"++"d"
2.1.f. fi "abcde" 'f' R = "abc"++"d" y
2.1.g. fi "xyzw" 'w' R = "xyz"

2.2. Dada la siguiente función

```
fa (a:[]) = [a]
```

fa (x:xs) = (fa (xs)) ++ "x"

Decir cuales de las siguientes afirmaciones son verdaderas:

2.2.a. fa "abcd" R = "abcx"
2.2.b. fa "abcx" R = "xxxx"
2.2.c. fa "a" R = "ax"
2.2.d. fa "xxxx" R = "xxxx"
2.2.e. fa "" R = ""
2.2.f. fa "" R = "x"
2.2.g. fa "xxxy" R = "y"

2.3. Definir Domino y Rango de la siguiente función
otro [] [] = []
otro [] [a] = []
otro (x:xs) [] = []
otro (x:xs) [a] = x:a

2.4. Definir recursivamente una función que retorne la longitud de una cadena de caracteres.

2.5. Escribir un programa funcional que determine la longitud de una lista cualquiera.

2.6. Dada una cadena s / long(s) >= n, implemente una función **carn** que retorne el n-ésimo carácter de s.

2.7. Escribir una función recursiva que sume todos los números menores de 3 que aparezcan en una lista de enteros

2.8. Escribir una función recursiva acumulativa que calcule el máximo elemento de la lista

2.9. Mostrar si el enésimo entero de una lista ingresada es par.

2.10. Definir recursivamente una función **cola** que devuelve los últimos n-1 caracteres de una cadena.

2.11. Definir recursivamente una función **concola** que concatena un carácter a una cadena.

2.12. Definir recursivamente una función reversa que invierta el orden de los elementos de una lista.

2.13. Definir recursivamente una función que devuelva la cantidad de 'a' en una cadena.

2.14. Definir recursivamente una función que devuelva la cantidad de 'la' en una cadena.

2.15. Escribir funciones que permitan duplicar los elementos de una lista y otra que permitan replicar los elementos de una lista dependiendo del número n que se pase como parámetro.

2.16. Escribir una función que rote a izquierda los elementos de una lista n posiciones. Dar una solución para que la función permita valores de n negativos (deberá rotar para la derecha).

2.17. Escriba una función **prom** que retorne el promedio de una secuencia de números almacenados en una lista.

2.18. Escribir un programa funcional que determine si un elemento es miembro de una lista.

2.19. Escribir un programa funcional que concatene dos listas.

2.20. Escribir un programa funcional que determine si una secuencia es subsecuencia de otra secuencia dada.

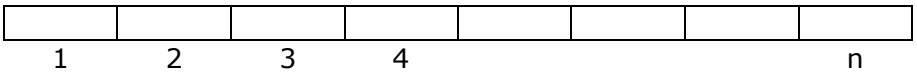
2.21. Escribir una función que determine si cada uno de los elementos de una cadena de enteros está contenida en otra. En la primera cadena no hay repetidos. Las coincidencias son correlativas: si m esta después de n en la primera cadena entonces m está después de n en la segunda cadena.
Ejemplo f [1,3,5] [1,2,3,4,5,5] = True

2.22. Escribir un programa que devuelva la serie de Fibonacci hasta él término enésimo.
Ejemplo: serie de Fibonacci = 1 1 2 3 5 8 13 …
fiboc 6 = [8,5,3,2,1,1]

2.23. Construir una función 'divide' que tome una lista 'xs' y una condición 'f' y devuelve (ys,zs) donde 'ys' = primeros elementos que cumplen la condición y 'zs' = resto de elementos.

Ejemplo: ?- divide (<3) [1..5]
([1,2],[3,4,5]).

2.24. En un barrio los niños juegan a la siguiente rayuela: se dibuja una hilera de n rectángulos



El jugador comienza posicionado en el rectángulo 0. En cada paso del juego puede saltar al rectángulo siguiente o a uno más allá.
Esto es, por ejemplo, que del 2 puede pasar al 3 o al cuatro y a ningún otro.
Escriba una función que compute la cantidad de caminos distintos que puede seguir cada jugador para llegar al rectángulo n partiendo del 1.

2.25. Escriba una función **palíndromo** que devuelva True o False si una cadena es igual a su reverso.

2.26. Escriba una función tal que dada una cadena de caracteres que contiene palabras separadas por blancos (exclusivamente) devuelva la primer palabra (hasta el primer blanco). No hay blancos iniciales.

2.27. Escriba una función tal que dada una cadena de caracteres que contiene palabras separadas por blancos devuelva la primer palabra. Pueden haber blancos iniciales. Ayuda: definir previamente una función que elimine los blancos iniciales.

2.28. Definir una función que cuente la cantidad de palabras que hay en una cadena de caracteres según se describió en los ejercicios anteriores.

2.29. Implementar en Haskell una función que dada una cadena que comprima los elementos duplicados consecutivos. Los mismos deben ser codificados en una lista de tuplas (N, E) donde N es el número de duplicaciones del E.

2.30. Definir una función que genere una lista de tuplas de tres elementos (i, j, k), donde el primer elemento sea múltiplo de 2, el segundo elemento sea múltiplo de 3 y el

tercer elemento sea el producto de ambos, con la condición de que la suma de los i y j sea múltiplo de 5.

3. Funciones de Orden Superior

3.1. Definir una función **aplicaSeq** para aplicar una función a todos los elementos de una secuencia

3.2. Definir filtrar que aplicada a un predicado y una lista produzca la sublista formada por los elementos de la lista para los que el predicado se evalúa a True, y definir otra función rechazar con el comportamiento contrario.

3.3. Definir una función **parList**, con una función de dos argumentos y dos listas como argumentos, que construya una lista con los resultados de aplicar la función del argumento a los elementos correspondientes (situados en las mismas posiciones) de ambas listas, permitiendo listas con distinta longitud.
p.e.: parList (+) [3,4] [2,2,1] = [5,6]

4. Cálculo con estructuras infinitas.

4.1. Definir una función **listasuc** que genere la lista infinita de los números siguientes a uno dado.

4.2. Con la función anterior y la función filtrar construir una lista con los 10 primeros múltiplos de 7.

4.3. Definir una función **rep** que genere la lista infinita resultante de la aplicación sucesiva de una función de un argumento a un cierto valor, e.d. tal que:
rep f x = [x, f x, f (f x), ...].

4.4. Con ayuda de esta función, definir funciones para calcular:
4.4.a. la lista de los múltiplos de 5,
4.4.b. la lista de potencias de 2,
4.4.c. la lista [True, False, True, False, ...],
4.4.d. la lista ['*','**','***','****',...].

4.5. Escribir la función **potencias**, tal que devuelva una lista infinita:
$[n^m, n^{m+1}, n^{m+2}, \ldots]$

Trabajo Práctico 6	*PROGRAMACIÓN LOGICA*

1. Lógica proposicional

1.1. Indicar si **s** se deriva de los programas siguientes utilizando resolución proposicional.

(a) $(p \wedge q \wedge r) \Rightarrow s$
(b) $(t \wedge w) \Rightarrow r$
(c) $(v \wedge r) \Rightarrow p$
(d) $v \Rightarrow w$
(e) q
(f) t
(g) v

(a) $(p \wedge q \wedge r) \Rightarrow s$
(b) $(m \wedge n) \Rightarrow r$
(c) $w \Rightarrow p$
(d) $(q \wedge r) \Rightarrow s$
(e) $z \Rightarrow q$
(f) $q \Rightarrow m$
(g) z
(h) n

1.2. Considere el siguiente programa en lógica proposicional usando notación Prolog.

```
(a)   t :- s, w, p, q.
(b)   u :- p, q, r, s, t.
(c)   u :- v, m.
(d)   u :- m.
(e)   t :- n.
(f)   v :- r.
(g)   w :- p, q.
(h)   p.
(i)   r.
(j)   q.
```

Construya un árbol de resolución SLD (arr-aba / izq-der) para el query **u**.
Muestre también que **¬t** se deriva del programa.

2. Lógica de Predicados

2.1. Determine si los siguientes pares de expresiones pueden unificarse. Si es posible, entonces dé el unificador más general; si no, de una breve explicación.

color(piolín, amarillo) - color(X, X).

color(sombrero(cartero), azul) - color(sombrero(X), Y)

r(f(X), b) - r(X, Y)

r(f(Y), X) - r(X, f(b))

r(f(Y),Y,X) - r(X,f(a),f(Z))

2.2. Dado el siguiente programa lógico, armar un árbol de derivación SLD para determinar si **hija (X, homero)** y **hijo (X, marge)** peden ser derivado del mismo. Indicar en cada paso cual es el unificador utilizado. Finalmente diga que valores puede tomar X.

```
(a)     hijo (X, Y) :- padre (Y, X), varon (X)
(b)     hija (X, Y) :- padre (Y, X), mujer (X)
(c)     padre (homero, bart)
(d)     padre (homero, lisa)
(e)     padre (homero, maggie)
(f)     varon (homero)
(g)     varon (bart)
(h)     mujer (lisa)
(i)     mujer (maggie)
```

Trabajo Práctico 7	*PROGRAMACIÓN LÓGICA CON PROLOG*

1. Ejercicios Preliminares

1.1. Convertir las siguientes frases en un programa de lógica proposicional.

Si Juan tiene hambre entonces come.
Si Juan esta borracho entonces esta alegre.
Si Juan esta cantando y riendo entonces esta borracho.
Si Juan esta comiendo y esta viendo TV entonces esta alegre.
Si Juan esta alegre y tomando entonces esta viendo TV.
Si Juan esta en el bar entonces esta tomando.
Si Juan esta cantando entonces esta alegre.
Si Juan esta alegre entonces esta en el bar.
Juan tiene hambre.
Juan esta cantando.

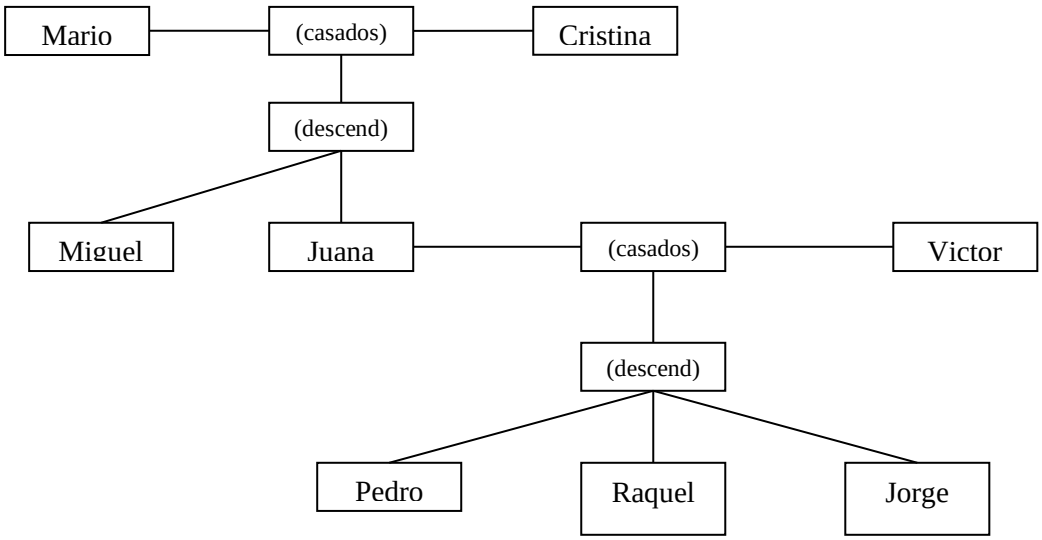
Determinar si es posible que el hecho de que Juan esta contento pueda derivarse del programa anterior.

1.2. Considerando las siguientes sentencias.

A Juan le agrada todo tipo de alimentos.
Las manzanas constituyen un alimento.
El pollo es un alimento.
Cualquier cosa que una persona coma y no lo mate es un alimento.
José se alimenta de ajíes picantes y aún esta vivo.
Mariana ingiere todo lo que come José.

- Traducir estas sentencias formuladas en Lógica de Predicados.
- Determinar si a Juan le agradan ajíes picantes.
- Determinar que comida ingiere Mariana.

1.3. Dado las siguientes relaciones.



- Utilizar las relaciones **casados**, **descendiente**, **varón** y **mujer** para programar el árbol genealógico, usando solamente aseveraciones.

- Con base en las aseveraciones programadas en el punto anterior añadir a la base de conocimiento reglas que puedan definir: **hijo**, **hija**, **madre**, **padre**.

1.4. Utilice la idea de estructura para programar un sistema pequeño de registro de personal. Puede registrar en el sistema a estudiantes de un curso, empleados de una empresa o miembros de un club.

1.5. Describir el proceso con que Prolog resolvería la meta presidente(X, Y) de acuerdo con la base de conocimiento siguiente.

```
miembro(becerril, pueblo_nuevo).
miembro(peña, pueblo_nuevo).
miembro(pacheco, villa_rica).
miembro(ramirez, pueblo_nuevo).
candidato(peña, pueblo_nuevo).
candidato(ramirez, pueblo_nuevo).
electo(peña).
electo(pacheco).
presidente(X,Y) :- miembro(X,Y), candidato(X,Y), electo(X).
```

2. El símbolo de corte "cut" !

```
progenitor(juan).
progenitor(alfredo).
progenitora(juana).
varon(juan).
varon(alfredo).
mujer(juana).
padre(X) :- progenitor(X), varon(X), !.
madre(X) :- progenitora(X), !, mujer(X).
```

1.1. Al utilizar la base de conocimiento anterior, indique que efecto tendrá la inclusión del símbolo de corte en la forma en que se responderá a las preguntas.

```
padre(X).
madre(X).
```

3. Acertijos Lógicos

3.1. El señor Pardo, el señor Castaño y el señor Blanco.

"Tres hombres se encuentran en la calle: el señor Pardo, el señor Castaño y el señor Blanco".

- ¿Se dan cuenta de que uno de nosotros va vestido de pardo, otro de castaño y otro de blanco? – pregunta el señor Pardo. – Sin embargo, ninguno lleva el traje del color de su nombre.
- Pues es verdad – dice el hombre de blanco.

¿De qué color va vestido cada uno?

3.2. En el negocios de usados.

Ante la dura situación económica, tres amigos fueron juntos a vender algunas de sus pertenencias a un negocio de usados. Juan, José y Mario (cuyos apellidos son Garcia, Perez y Rossi, pero no es ese orden), vendieron un televisor, una notebook y una lámpara, habiendo cobrado $150, $200 y $300 (nuevamente, no en especial orden). Usando

las pistas que siguen, obtener el nombre y apellido, que vendió y cuanto cobró cada uno.

- La notebook fue el ítem más barato.
- Juan no vendió el televisor.
- José cobró menos que Perez (quién no vendió la lámpara).
- Rossi cobró $300.

Nota: no se garantiza la veracidad de ninguna presunción que no esté expresamente dicha en el enunciado, por ejemplo que un televisor vale más que una lámpara, etc.

3.3. Las tres mujeres y sus vestidos.

"Tres mujeres. Mara, Cora y Rita llevan ropas en azul, verde, gris, rojo y amarillo. Ninguna viste rojo y amarillo. Cada una tiene un traje de dos piezas en dos colores. Salvo Cora, ninguna mujer viste el mismo color que la otra. Rita tiene algo azul. Mara viste amarillo pero no verde. Cora viste verde pero no azul o gris. Rita no viste ninguno de los que tiene Mara. Encontrar los colores que viste cada mujer."

3.4. Jarros

Se dispone de dos jarros, uno con una capacidad de 3 litros y otro con una capacidad de 4 litros. Ninguno de los dos tiene marcas que permitan medir cantidades que no sean las de sus propias capacidades. Existe una canilla que permite llenar los jarros con agua y un sumidero donde se puede vaciar los mismos. El problema consiste en encontrar cuál es la secuencia de movimientos de llenado, vaciado y trasvase que permitan obtener exactamente dos litros de agua en el jarro de 4 litros.

4. La recurrencia en Prolog

4.1. Escriba reglas adicionales a la siguiente base de conocimiento para poder definir ancestros, utilizando una definición recurrente.

```
padres(jaime, juan, elena).
padres(juan, guillermo, doris).
padres(elena, efraín, alma).
padres(guillermo, jose, flor).
padres(doris, alberto, sonia).
padres(efrain, crispin, claudia).
padres(alma, ruperto, alicia).
```

Todas las afirmaciones anteriores son de la forma **padres(X,Y,Z)**, que se lee "los padres de X son Y y Z".

4.2. Escriba un programa que calcule el factorial de un número.
4.3. Implemente un predicado que determine el término de fibonacci correspondiente a su orden.
4.4. Escriba los 15 primeros términos de fibonacci.
4.5. La expresión matemática n C r se obtiene como sigue: dividir el factorial de n entre el producto del factorial de r y el factorial de (n-r). Programa un procedimiento para calcular n C r.
4.6. Escriba un programa para calcular el valor de Xn , donde X y n son enteros.

5. Optimización de la última llamada.

5.1. Escriba un programa recursivo que utilice la optimización de la última llamada que acepte un número como entrada y pueda finalizar de 2 formas. Se comenzará multiplicando el número por si mismo una y otra vez hasta alcanzar 81 o alcanzar

un número mayor que 100. Si se alcanza 81 se imprimirá "Si", si se excede 100 se imprimirá "No".

6. Tratamiento de listas.

6.1. Definir un procedimiento que permita verificar si un elemento pertenece o no a una lista.
6.2. Definir un procedimiento que permita concatenar dos listas, es decir que los elementos de la segunda lista deben ir a continuación de la primera.
6.3. Definir un procedimiento que obtenga el último elemento de la lista.
6.4. Definir un procedimiento que permita determinar la cantidad de elementos en una lista.
6.5. Definir un procedimiento que genere una lista de enteros consecutivos a partir de un valor mínimo y un máximo.
6.6. Definir un procedimiento que permita invertir el orden en que se encuentran los elementos de una lista.
6.7. Definir un procedimiento que permita eliminar los elementos que sean iguales al que estamos considerando.
6.8. Implemente un predicado que establezca si dos listas comparten como mínimo un elemento.
6.9. Definir un procedimiento que imprima el contenido de una matriz.

7. Ejercicios tomados de la guía de Algoritmos y Estructuras de Datos.

7.1. Elabore un algoritmo que permita obtener el máximo común divisor de dos números.
7.2. Construya un algoritmo capaz de encontrar todas las cifras de tres dígitos que cumplan con la condición de que la suma de los cubos de sus dígitos sea igual al número que la cifra representa.
7.3. Se desea determinar si un estudiante aprobó una materia. Para ello se le toman 5 exámenes (E1, E2, E3, E4, E5). Se considera aprobada la materia a aquellos alumnos que aprobaron los 5 exámenes: deberán rendir un examen de complemento aquellos que aprobaron 3 exámenes, siempre que uno de estos tres sea el último. Tendrán la posibilidad de volver a rendir el último examen aquellos alumnos que no lo aprobaron, pero aprobaron por lo menos el 3ro y el 4to.
7.4. Diseñar un algoritmo para acceder al k-ésimo elemento de una lista. Si la lista está vacía o si el valor de k está fuera de rango del índice de la lista, invocar al procedimiento ERROR. En cualquier otro caso, retornar el valor solicitado.