



UANL

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN



FIME

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica
y Eléctrica

Examen Ordinario

Nombre: Ranfery Josua Peregrina Morales

Matrícula: 1924910

Semestre: Décimo

Materia: Temas Selectos de IA

Docente: Raymundo Said Zamora Pequeño

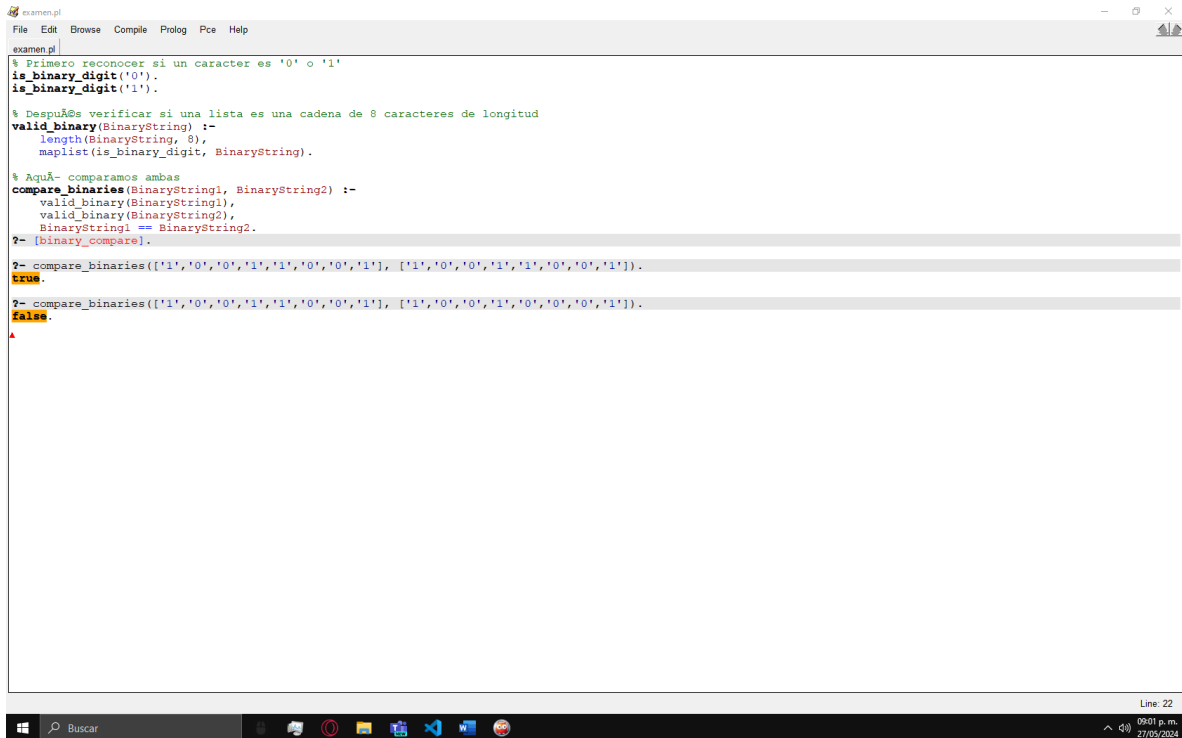
Grupo: 001

Fecha: 27/05/2024

Periodo: *Enero – Junio 2024*



Pregunta 1: Un programa en PROLOG que permita comparar 2 strings binarios de 8 dígitos.



```
examen.pl
File Edit Browse Compile Prolog Pce Help
examen.pl
% Primero reconocer si un caracter es '0' o '1'
is_binary_digit('0').
is_binary_digit('1').

% Después verificar si una lista es una cadena de 8 caracteres de longitud
valid_binary(BinaryString) :-
    length(BinaryString, 8),
    maplist(is_binary_digit, BinaryString).

% Aquí- comparamos ambas
compare_binaries(BinaryString1, BinaryString2) :-
    valid_binary(BinaryString1),
    valid_binary(BinaryString2),
    BinaryString1 == BinaryString2.

?- [binary_compare].
?- compare_binaries(['1','0','0','1','1','0','0','1'], ['1','0','0','1','1','0','0','1']).
true.
?- compare_binaries(['1','0','0','1','1','0','0','1'], ['1','0','0','1','0','0','0','1']).
false.
^
```

% Primero reconocer si un caracter es '0' o '1'

is_binary_digit('0').

is_binary_digit('1').

% Después verificar si una lista es una cadena de 8 caracteres de longitud

valid_binary(BinaryString) :-

length(BinaryString, 8),

maplist(is_binary_digit, BinaryString).

% Aquí- comparamos ambas

compare_binaries(BinaryString1, BinaryString2) :-

valid_binary(BinaryString1),

valid_binary(BinaryString2),

BinaryString1 == BinaryString2.

?- [binary_compare].

```
?- compare_binaries(['1','0','0','1','1','0','0','1'], ['1','0','0','1','1','0','0','1']).
```

```
?- compare_binaries(['1','0','0','1','1','0','0','1'], ['1','0','0','1','0','0','0','1']).
```

No sé por qué se pegó sin formato ni colores, sólo como texto, se ve algo feo pero se entiende. Sobre todo, porque era un programa sencillo... sospechosamente sencillo.

De todas formas, no escribí el código directamente en el Edit de Prolog porque no me daba confianza. Lo hice en un editor de texto y lo pasé a Prolog.

El código más legible luce así:

```
% Primero reconocer si un caracter es '0' o '1'
is_binary_digit('0').
is_binary_digit('1').

% Después verificar si una lista es una cadena de 8 caracteres de longitud
valid_binary(BinaryString) :-
    length(BinaryString, 8),
    maplist(is_binary_digit, BinaryString).

% Aquí comparamos ambas
compare_binaries(BinaryString1, BinaryString2) :-
    valid_binary(BinaryString1),
    valid_binary(BinaryString2),
    BinaryString1 == BinaryString2.
?- [binary_compare].
```

Pregunta 2: Desarrollar 3 generaciones de una población:

Estoy dedicando 1 bit al signo:
0 si el número recibido es positivo.
1 si el bit recibido es negativo.

Estoy dedicando 9 bits a la parte entera
Para poder representar desde 0, hasta 256
Y así poder cubrir los: 170

Estoy utilizando 11 bits a la parte decimal.
Para poder representar hasta 1024
Y así poder cubrir los posibles: 999 de posible decimal.
Usando un total de 21 caracteres para representar cada individuo

Y cada cromosoma tiene 2 individuos así que serían 42 caracteres por individuo
Por ejemplo: [-1,1] está representado de esta forma:

[illegible]

```
[1] "Individuo después de la mutación en posición 4 : 00010010110000000000000000000001
00000000000"
[1] "Calificación del individuo: 0000000001000000000000000110010000000000000 es: 38"
[1] "Calificación del individuo: 010000000000000000010000000000100000000000 es: 39"
[1] "Calificación del individuo: 10100000010000000000000010101010000000000000 es: 35"
[1] "Calificación del individuo: 00010010110000000000000000000000100000000000 es: 37"
[1] 38 39 35 37
Puntuación después de mutar: 149
```

```
[1] "=====
```

Generación: 2

```
[1] "000000000100000000000000011001000000000000"
[2] "0100000000000000000000000100000000000000"
[3] "101000000100000000000000101010100000000000"
[4] "0001001011000000000000000000000100000000000"
[1] "Calificación del individuo: 000000000100000000000000011001000000000000 es: 38"
[1] "Calificación del individuo: 010000000000000000010000000000100000000000 es: 39"
[1] "Calificación del individuo: 101000000100000000000000101010100000000000 es: 35"
[1] "Calificación del individuo: 00010010110000000000000000000000100000000000 es: 37"
[1] 38 39 35 37
```

Puntuación inicial: 149

```
[1] "Mejores individuos seleccionados:"
[1] "0100000000000000000000000100000000000000"
[2] "000000000100000000000000011001000000000000"
[3] "0001001011000000000000000000000100000000000"
[4] "101000000100000000000000101010100000000000"
[1] "Cruzando: 010000000000000000010000000000100000000000 y 00000000010000000000000001
100100000000000000"
[1] "Resulta en hijos: 010000000000000000010000110010000000000000 y 000000000100000000
00000000000001000000000000"
[1] "Cruzando: 010000000000000000010000000000100000000000 y 00010010110000000000000000
000001000000000000"
[1] "Resulta en hijos: 010000000000000000010000000000100000000000 y 000100101100000000
00000000000001000000000000"
[1] "Nueva población generada:"
[1] "01000000000000000000000001000011001000000000000"
[2] "000000000010000000000000000000000100000000000"
[3] "010000000000000000000000010000000000010000000000"
[4] "000100101100000000000000000000000100000000000"
```

```
[1] "Individuo después de la mutación en posición 1 : 11000000000000000000100001100100
000000000000"
[1] "Individuo después de la mutación en posición 2 : 0100000001000000000000000000000001
000000000000"
[1] "Individuo después de la mutación en posición 3 : 0110000000000000000010000000000001
000000000000"
[1] "Individuo después de la mutación en posición 4 : 0000001011000000000000000000000001
000000000000"
[1] "Calificación del individuo: 110000000000000000001000011001000000000000 es: 36"
[1] "Calificación del individuo: 01000000010000000000000000000000100000000000 es: 39"
[1] "Calificación del individuo: 01100000000000000000100000000000100000000000 es: 38"
[1] "Calificación del individuo: 00000010110000000000000000000000100000000000 es: 38"
[1] 36 39 38 38
```

Puntuación después de mutar: 151

```
[1] "=====
```

Generación: 3

```
[1] "11000000000000000000000001000011001000000000000"
[2] "0100000001000000000000000000000001000000000000"
[3] "0110000000000000000000000100000000000100000000000"
[4] "00000010110000000000000000000000010000000000000"
[1] "Calificación del individuo: 11000000000000000000000001000011001000000000000 es: 36"
[1] "Calificación del individuo: 010000000100000000000000000000000100000000000 es: 39"
[1] "Calificación del individuo: 01100000000000000000100000000000100000000000 es: 38"
[1] "Calificación del individuo: 000000101100000000000000000000000100000000000 es: 38"
[1] 36 39 38 38
```

Puntuación inicial: 151

```
[1] "Mejores individuos seleccionados:"
[1] "0100000001000000000000000000000001000000000000"
[2] "0110000000000000000000000100000000000100000000000"
[3] "00000010110000000000000000000000010000000000000"
```


Evolución de la Puntuación en Cada Iteración

