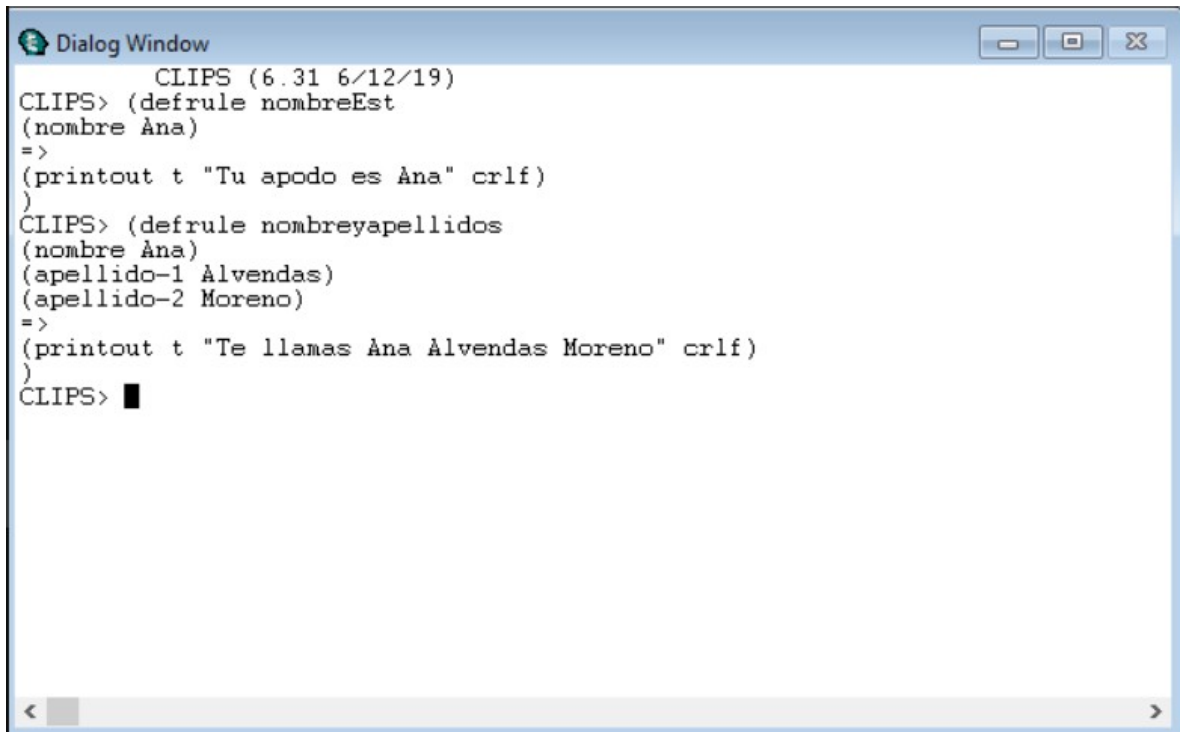


UNIVERSIDAD TECNOLÓGICA DE PANAMA
CENTRO REGIONAL DE PANAMA OESTE
FACULTAD DE INGENIERIA DE SISTEMAS COMPUTACIONALES
LICENCIATURA EN INGENIERIA EN SISTEMAS Y COMPUTACION
TALLER – CLIPS

PROBLEMA 1:

1. Arranque CLIPS desde su terminal.
2. Abra las ventanas Agenda y Hechos. Para ello, elija **Facts Window** y **Agenda Window** de la opción **Window** que aparece en el menú principal.
3. Introduzca las siguientes reglas:

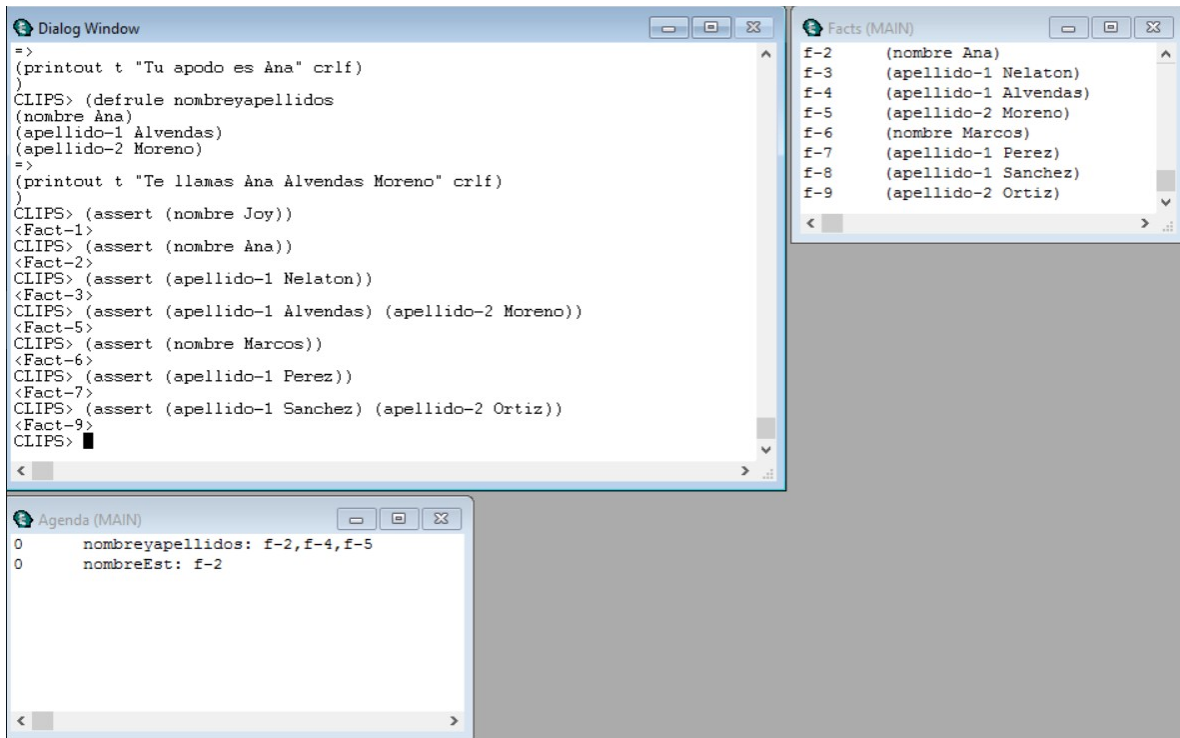
```
(defrule nombreEst
(nombre Ana)
=>
(printout t "Tu apodo es Ana" crlf)
)
(defrule nombreyapellidos
(nombre Ana)
(apellido-1 Alvendas)
(apellido-2 Moreno)
=>
(printout t "Te llamas Ana Alvendas Moreno" crlf)
)
```



```
CLIPS (6.31 6/12/19)
CLIPS> (defrule nombreEst
(nombre Ana)
=>
(printout t "Tu apodo es Ana" crlf)
)
CLIPS> (defrule nombreyapellidos
(nombre Ana)
(apellido-1 Alvendas)
(apellido-2 Moreno)
=>
(printout t "Te llamas Ana Alvendas Moreno" crlf)
)
CLIPS> █
```

4. Introduzca ahora los siguientes hechos

```
(assert (nombre <ponga su nombre de pila>))
(assert (nombre Ana))
(assert (apellido-1 <ponga su primer apellido>))
(assert (apellido-1 Alvendas) (apellido-2 Moreno))
(assert (Introduza ahora otros hechos))
```

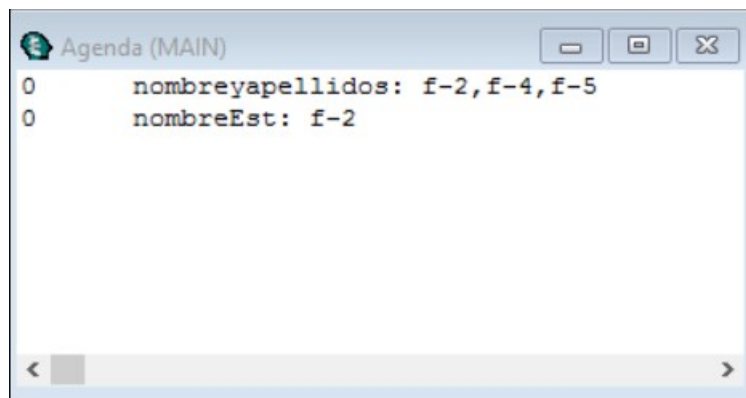


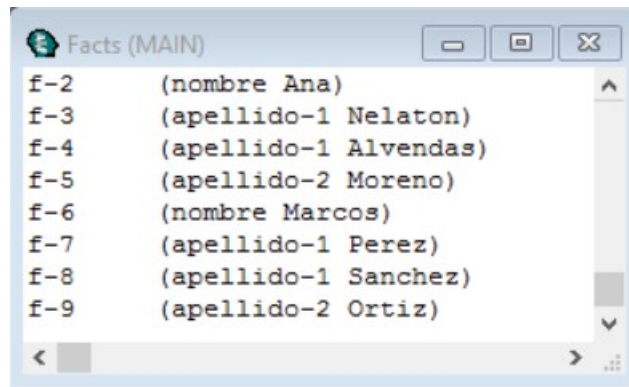
5. ¿Se ha activado alguna regla? ¿En qué orden se han activado?

Si, se han activado las reglas `nombreyapellidos` y `nombreEst`.

`NombreEst` se ha activado en el `fact-2`, correspondiendo a la inserción de el nombre Ana.

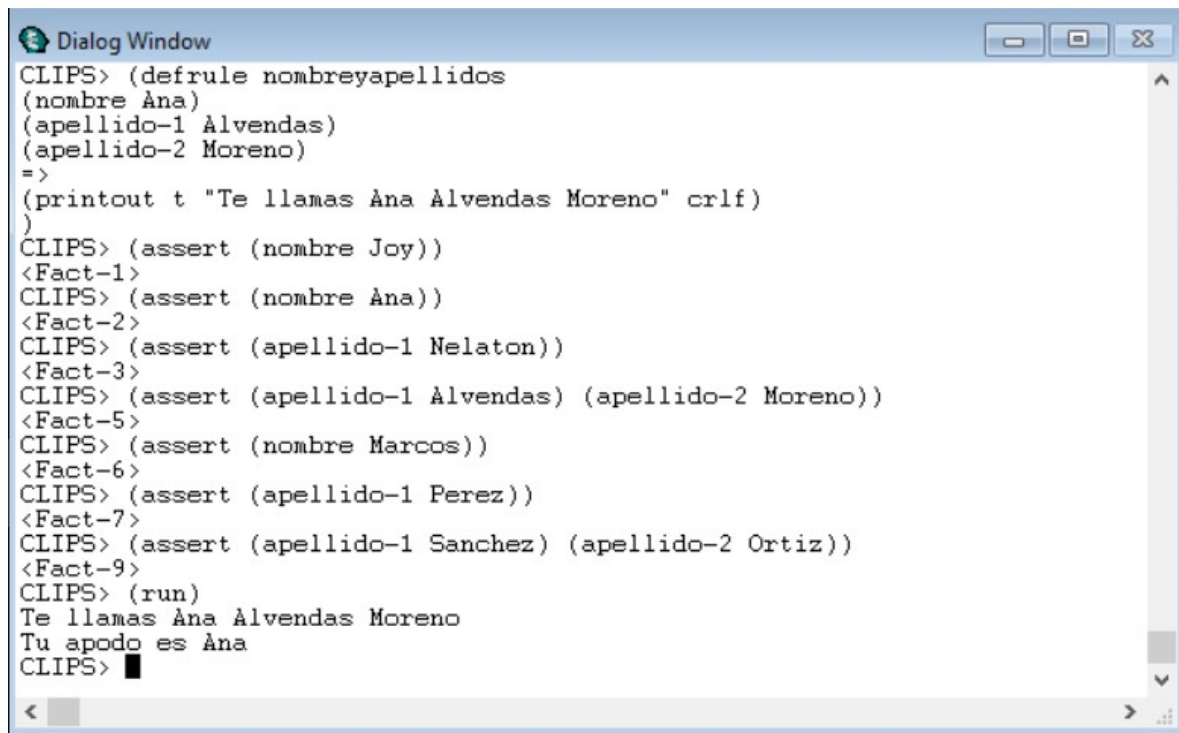
`Nombreyapellidos` se ha activado en el `fact-2`, correspondiendo a la inserción del nombre Ana y en los facts 4 y 5 correspondiendo a la inserción del apellido-1: Alvendas (`fact-4`) y del apellido-2: Moreno (`fact-5`)





6. Comience el ciclo de ejecución. Para ello, escriba (run). ¿Qué reglas se han ejecutado? ¿En qué orden lo han hecho? ¿Por qué cree que se han ejecutado en ese orden?

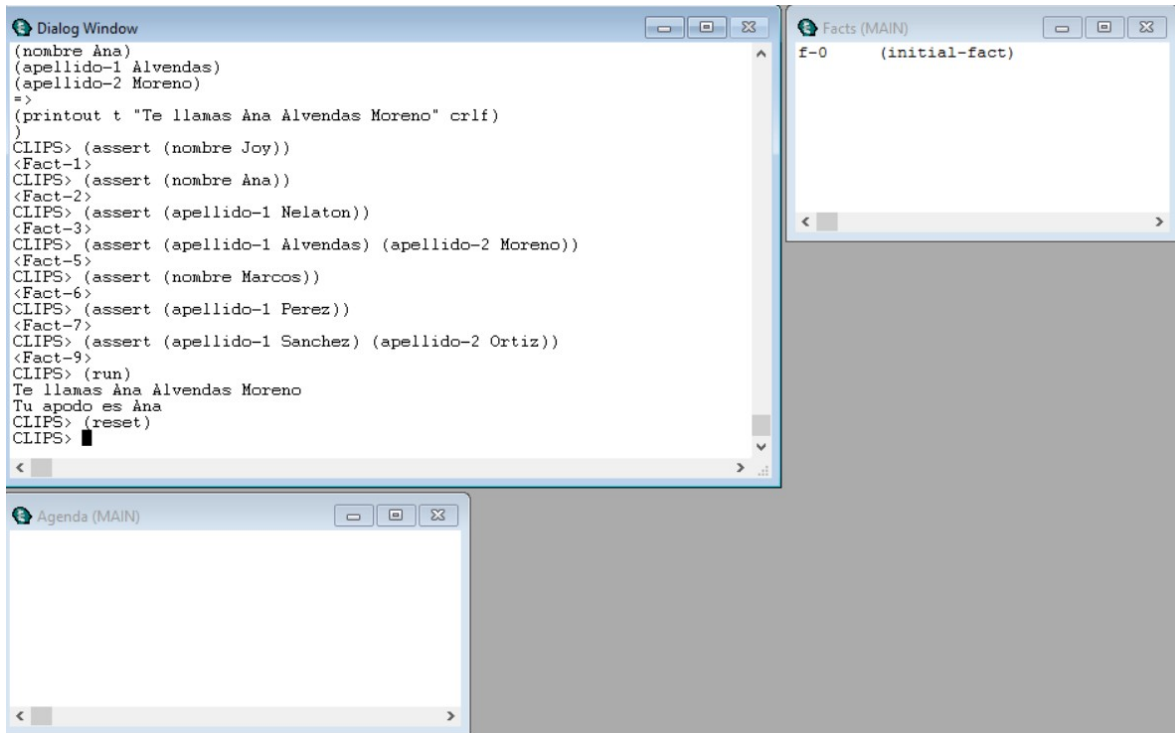
Se han ejecutado las reglas nombreyapellidos y nombreEst, en ese orden. Pienso que se han ejecutado en ese orden debido a que la regla nombreyapellidos tiene tres parametros, en cambio la regla nombreEst solo tiene un parametro.



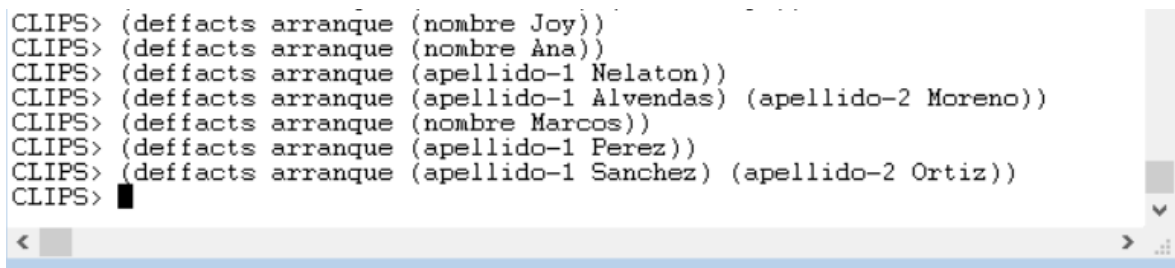
7. Reinicie el sistema con (reset). ¿Qué hubiera pasado si reiniciamos con (clear) en lugar de con (reset)? Observación: En el menú Execution existen, entre otras, las opciones **Run**, **Reset** y **Clear**, que evitan tener que introducir directamente estos mandatos.

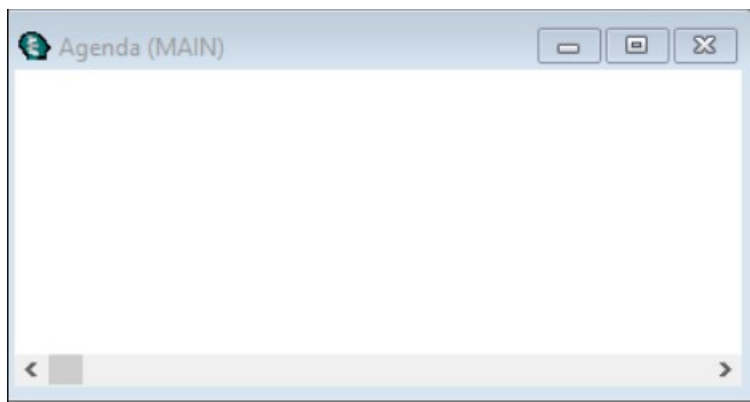
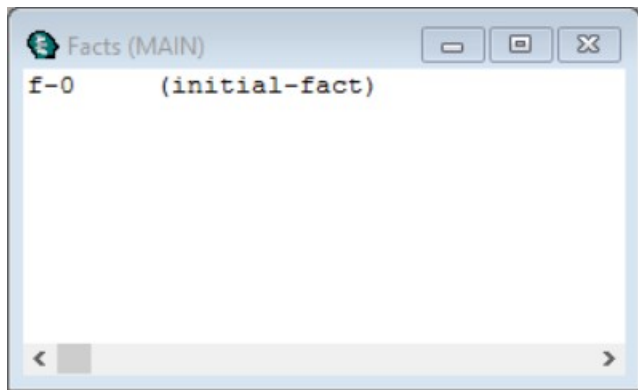
Reset: Elimina sólo los hechos, no las reglas, anulando la agenda y añadiendo los elementos definidos por defecto o iniciales.

Clear: Elimina todos los hechos y reglas almacenados en memoria, equivalente a cerrar y rearrancar CLIPS.



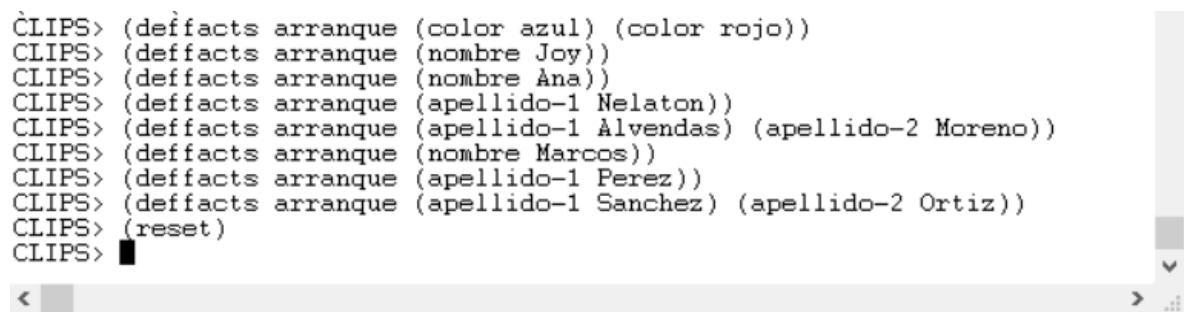
8. Introduzca los hechos anteriores utilizando el mandato (deffacts). ¿Se activa alguna regla ahora? ¿Cuántos hechos hay en la ventana de hechos? ¿Por qué?

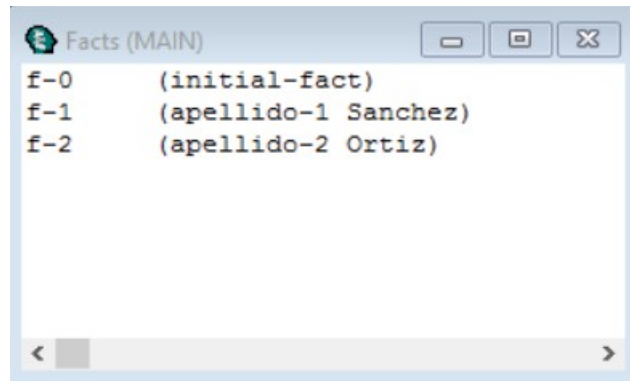




No hay hechos en la sección de facts, y no se han activado reglas, esto se debe a que se requiere de un reset para cargar hechos cargados utilizando deffacts, estos a su vez luego podrían activar reglas.

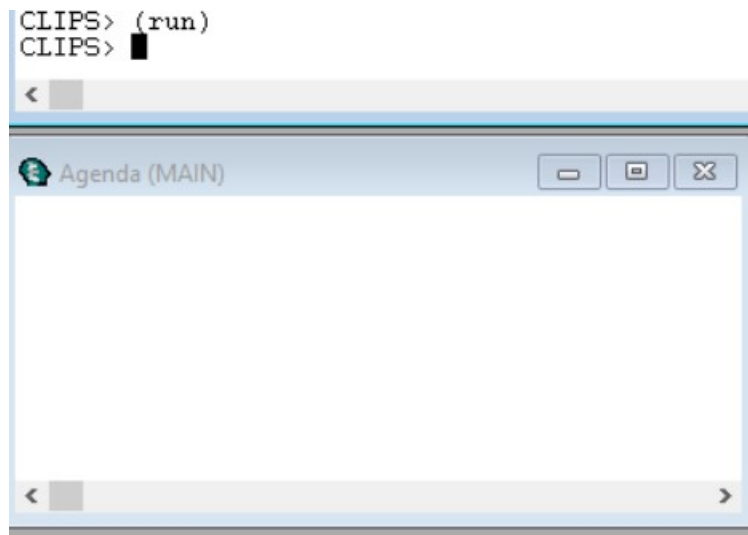
9. Reinicie con (reset) y ejecute con (run) y comente los resultados.



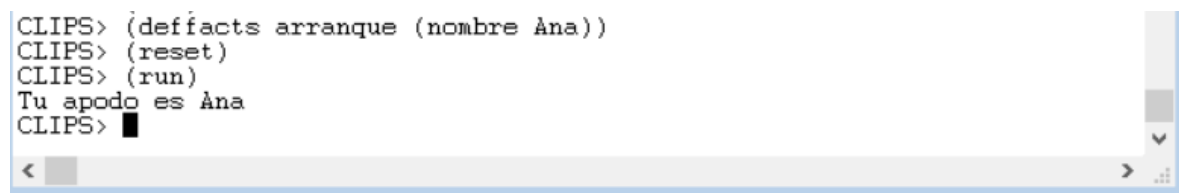


Al ejecutar el reset se observa que se cargan los hechos, apellido-1 Sanchez y apellido-2 Ortiz. Pienso que esto ocurre dado que fueron los últimos hechos cargados, y cuando utilizamos deffacts en la carga de los hechos anteriores, llamamos al deffact del mismo modo (arranque)

Al ejecutar run no se activan reglas, ya que los hechos cargados no activan las reglas previamente establecidas.



Para probar esta teoría procedimos a cargar nuevamente un hecho (nombre Ana) utilizando deffacts para luego realizar un reset y posteriormente un run, el resultado fue que tras esto la regla nombreEst si ha sido activada.



PROBLEMA 2: Nuestro primer SE o SI con CLIPS

A continuación se verá un ejemplo sencillo de un SE médico que presenta cinco reglas.

```
(defrule dar-digital "regla1"
```

```
(riesgo ?nombre infarto)
```

```
(anterior ?nombre infarto)
```

```
=>
```

```
(assert (dar ?nombre digital))
```

```
(printout t "dar a " ?nombre "digital." crlf))
```

```
(defrule riego-infarto "regla2"
```

```
(dolor ?nombre lado-izquierdo)
```

```
(alta ?nombre presion-arterial)
```

```
=>
```

```
(assert (riesgo ?nombre infarto))
```

```
(printout t ?nombre "corre riesgo de infarto." crlf))
```

```
(defrule alta-presion-arterial "regla3"
```

```
(alta ?nombre iop)
```

```
=>
```

```
(assert (alta ?nombre presion-arterial))
```

```
(printout t ?nombre "tiene la presion arterial alta." crlf))
```

```
(defrule esclerotico "regla4"
```

```
(paciente ?nombre muy-grueso fumador)
```

```
=>
```

```
(assert (propenso ?nombre a esclerosis))
```



```
(printout t ?nombre "es propenso a la esclerosis." crlf))
```

```
(defrule dar-digital2 "regla5"
```

```
(propenso ?nombre a esclerosis)
```

```
(alta ?nombre iop)
```

```
=>
```

```
(assert (dar ?nombre digital))
```

```
(printout t "dar a " ?nombre "digital." crlf))
```

```
(defacts estado-inicial
```

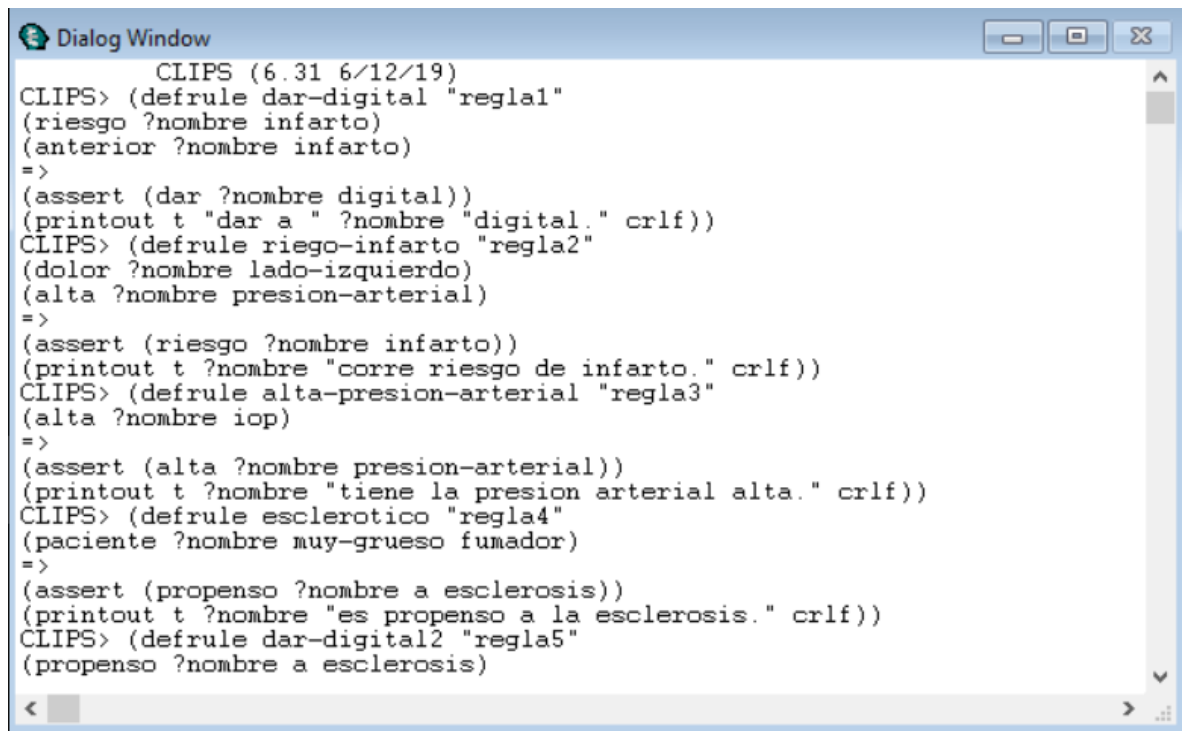
```
(alta Jose-Martinez iop)
```

```
(anterior Jose-Martinez infarto)
```

```
(dolor Jose-Martinez lado-izquierdo)
```

```
(paciente Juan-Lopez asmatico)
```

```
(paciente Jose-Martinez muy-grueso fumador))
```



```
CLIPS (6.31 6/12/19)
CLIPS> (defrule dar-digital "regla1"
(riesgo ?nombre infarto)
(anterior ?nombre infarto)
=>
(assert (dar ?nombre digital))
(printout t "dar a " ?nombre "digital." crlf))
CLIPS> (defrule riesgo-infarto "regla2"
(dolor ?nombre lado-izquierdo)
(alta ?nombre presion-arterial)
=>
(assert (riesgo ?nombre infarto))
(printout t ?nombre "corre riesgo de infarto." crlf))
CLIPS> (defrule alta-presion-arterial "regla3"
(alta ?nombre iop)
=>
(assert (alta ?nombre presion-arterial))
(printout t ?nombre "tiene la presion arterial alta." crlf))
CLIPS> (defrule esclerotico "regla4"
(paciente ?nombre muy-grueso fumador)
=>
(assert (propenso ?nombre a esclerosis))
(printout t ?nombre "es propenso a la esclerosis." crlf))
CLIPS> (defrule dar-digital2 "regla5"
(propenso ?nombre a esclerosis)
```

```

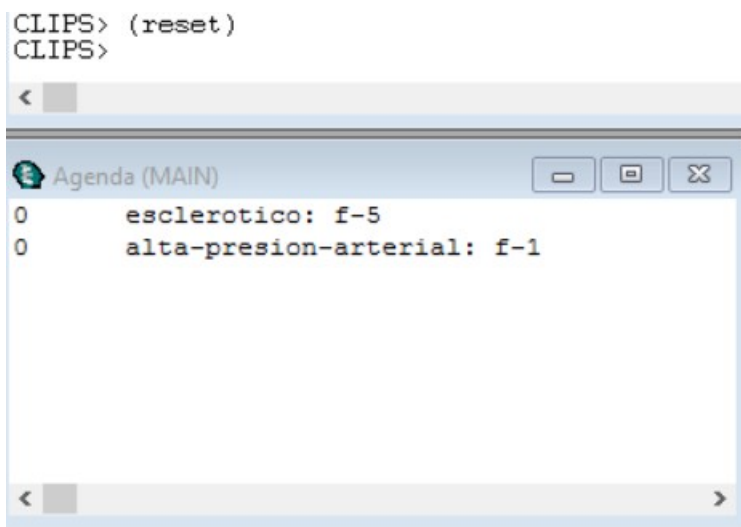
(assert (propenso ?nombre a esclerosis))
(printout t ?nombre "es propenso a la esclerosis." crlf))
CLIPS> (defrule dar-digital2 "regla5"
(propenso ?nombre a esclerosis)
(alta ?nombre iop)
=>
(assert (dar ?nombre digital))
(printout t "dar a " ?nombre "digital." crlf))
CLIPS> (deffacts estado-inicial
(alta Jose-Martinez iop)
(anterior Jose-Martinez infarto)
(dolor Jose-Martinez lado-izquierdo)
(paciente Juan-Lopez asmatico)
(paciente Jose-Martinez muy-grueso fumador))
CLIPS> █

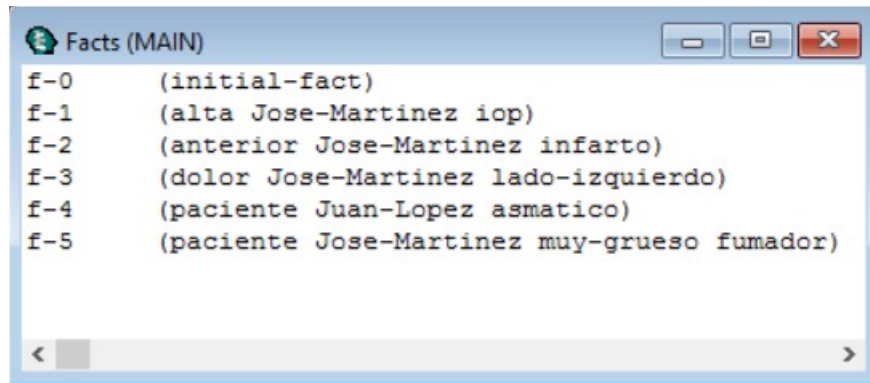
```

Para ejecutar este programa de prueba, abra CLIPS y cargue el fichero que contiene las definiciones de hechos y reglas.

1. Arranque CLIPS desde su terminal.
2. Abra las ventanas Agenda y Hechos. Para ello, elija **Facts Window** y **Agenda Window** de la opción **Window** que aparece en el menú principal.
3. Cargue el fichero que contiene el Sistema Experto anterior
4. Reinicie el sistema con (reset)
5. ¿Se ha activado alguna regla?

Se han activado las reglas esclerotico (fact-5) y alta-presion-arterial (f-1)





Esto corresponde a los hechos anteriormente insertados usando deffacts.

```
CLIPS> (defacts estado-inicial
(alta Jose-Martinez iop)
(anterior Jose-Martinez infarto)
(dolor Jose-Martinez lado-izquierdo)
(paciente Juan-Lopez asmatico)
(paciente Jose-Martinez muy-grueso fumador))
CLIPS> (reset)
-----
```

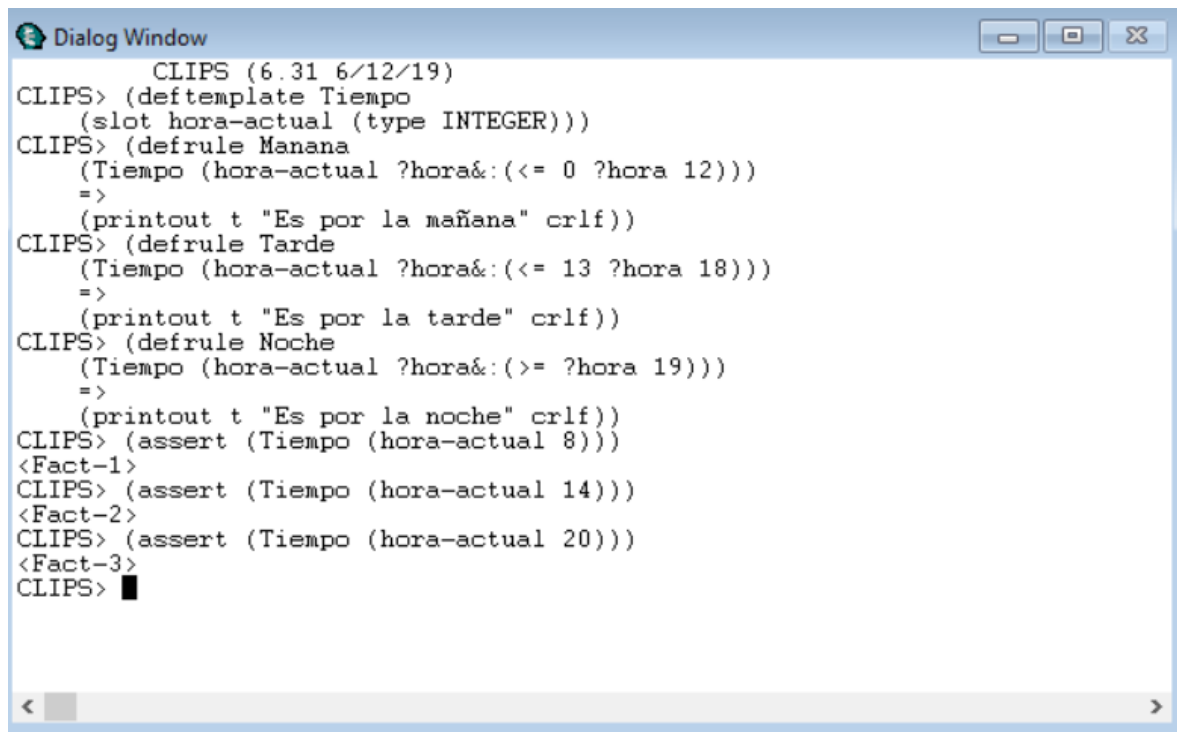
6. Comience el ciclo de ejecución. Para ello, escriba (run). ¿Qué reglas se han ejecutado? ¿En qué orden lo han hecho? ¿Por qué cree que se han ejecutado en ese orden?

```
CLIPS> (run)
Jose-Martinezes propenso a la esclerosis.
dar a Jose-Martinezdigital.
Jose-Martineztiene la presion arterial alta.
Jose-Martinezcorre riesgo de infarto.
dar a Jose-Martinezdigital.
CLIPS>
```

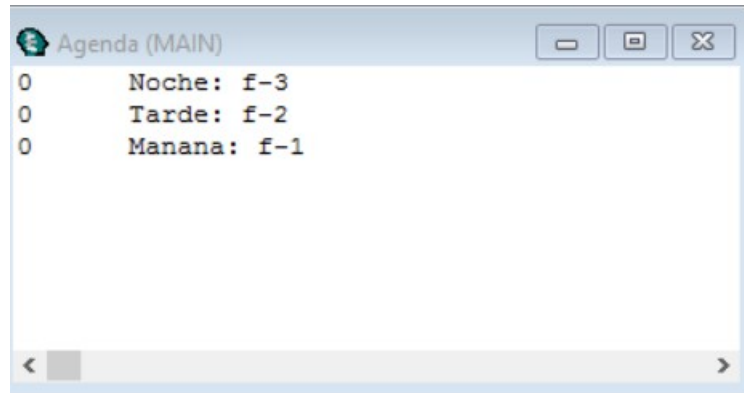
Se han activado las reglas en el siguiente orden: regla4 (esclerotico), regla5 (dar-digital2), regla3 (presion-arterial), regla2(riego-infarto) y la regla1 (dar-digital).

Se han ejecutado en ese orden ya que el ultimo hecho insertado mediante deffacts fue paciente Jose-Martinez muy-grueso fumador. Esto activa la regla4 (esclerotico) y da pie a la activación de las reglas siguientes que tienen dependencia la una de la otra (primero se debe cumplir una para que la otra se ejecute).

7. Desarrolle un pequeño sistema experto con 3 ó 4 reglas del estilo del que ha visto en el ejemplo anterior. Introduzca una serie de hechos para comprobar que funciona correctamente.



```
CLIPS (6.31 6/12/19)
CLIPS> (deftemplate Tiempo
  (slot hora-actual (type INTEGER)))
CLIPS> (defrule Manana
  (Tiempo (hora-actual ?hora&:(<= 0 ?hora 12)))
  =>
  (printout t "Es por la mañana" crlf))
CLIPS> (defrule Tarde
  (Tiempo (hora-actual ?hora&:(<= 13 ?hora 18)))
  =>
  (printout t "Es por la tarde" crlf))
CLIPS> (defrule Noche
  (Tiempo (hora-actual ?hora&:(>= ?hora 19)))
  =>
  (printout t "Es por la noche" crlf))
CLIPS> (assert (Tiempo (hora-actual 8)))
<Fact-1>
CLIPS> (assert (Tiempo (hora-actual 14)))
<Fact-2>
CLIPS> (assert (Tiempo (hora-actual 20)))
<Fact-3>
CLIPS> █
```



```
Agenda (MAIN)
0      Noche: f-3
0      Tarde: f-2
0      Manana: f-1
```

Al cargar los hechos se observan que se activan las reglas, noche, tarde y mañana, en ese orden, correspondiendo al orden inverso de inserción.

Al ejecutar run tenemos la salida en el orden anteriormente citado.

```
CLIPS> (run)
Es por la noche
Es por la tarde
Es por la mañana
CLIPS> █
```

BUENA SUERTE!!!

CLIPS - Programación Basada en Reglas

<https://www.monografias.com/docs114/clips-a-programacion-basada-reglas/clips-a-programacion-basada-reglas3.shtml>