

```
1 (defrule progenitor
2 (padre-de ?padre ?hijo)
3 (marido-de ?padre ?madre)
4 =>
5 (assert (progenitor-de ?madre ?hijo))
6 )
7 (defrule esposa
8 (marido-de ?hombre ?mujer)
9 =>
10 (assert (esposa-de ?mujer ?hombre))
11 (printout t ?mujer " es la esposa de " ?hombre crlf)
12 )
13 (defrule padre
14 (progenitor-de ?padre ?hijo)
15 (hombre ?padre)
16 =>
17 (assert (padre-de ?padre ?hijo))
18 (printout t ?padre " es padre de " ?hijo crlf)
19 )
20 (defrule madre
21 (progenitor-de ?madre ?hijo)
22 (mujer ?madre)
23 =>
24 (assert (madre-de ?madre ?hijo))
25 (printout t ?madre " es madre de" ?hijo crlf)
26 )
27 ; ----- Abuelos
28 (defrule abuelos
29 (progenitor-de ?padre ?hijo)
30 (progenitor-de ?hijo ?nieto)
31 =>
32 (assert (abuelos-de ?padre ?nieto))
33 )
34 (defrule abuelo
35 (abuelos-de ?padre ?nieto)
36 (hombre ?padre)
37 =>
38 (assert (abuelo-de ?padre ?nieto))
39 (printout t ?padre " es el abuelo de " ?nieto crlf)
40 )
41 (defrule abuela
42 (abuelos-de ?madre ?nieto)
43 (mujer ?madre)
44 =>
45 (assert (abuelo-de ?madre ?nieto))
46 (printout t ?madre " es el abuela de " ?nieto crlf)
47 )
48 ;----- Hermanos ---
49 (defrule hermanos-padre
50 (padre-de ?padre ?hijo1)
51 (padre-de ?padre ?hijo2)
52 (test (neq ?hijo1 ?hijo2))
53 =>
54 (assert (hermanos ?hijo1 ?hijo2))
55 )
56 (defrule hermanos-madre
57 (madre-de ?madre ?hijo1)
58 (madre-de ?madre ?hijo2)
59 (test (neq ?hijo1 ?hijo2))
60 =>
```

```
61 (assert (hermanos ?hijo1 ?hijo2))
62 )
63 (defrule hermano
64 (hermanos ?hijo1 ?hijo2)
65 (hombre ?hijo1)
66 =>
67 (assert (hermano-de ?hijo1 ?hijo2))
68 (printout t ?hijo1 " es hermano de " ?hijo2 crlf)
69 )
70 (defrule hermana
71 (hermanos ?hijo1 ?hijo2)
72 (mujer ?hijo1)
73 =>
74 (assert (hermana-de ?hijo1 ?hijo2))
75 (printout t ?hijo1 " es hermana de " ?hijo2 crlf)
76 )
77 ; ----- Tios -----
78 (defrule tios
79 (progenitor-de ?padre ?hijo)
80 (hermanos ?padre ?hermano)
81 =>
82 (assert (tios ?hermano ?hijo))
83 )
84 (defrule tio
85 (tios ?tio ?sobrino)
86 (hombre ?tio)
87 =>
88 (assert (tio ?tio ?sobrino))
89 (printout t ?tio " es tio de " ?sobrino crlf)
90 )
91 (defrule tia
92 (tios ?tia ?sobrino)
93 (mujer ?tia)
94 =>
95 (assert (tia-de ?tia ?sobrino))
96 (printout t ?tia " es tia de " ?sobrino crlf)
97 )
98 (defrule sobrino
99 (tios ?tios ?sobrino)
100 (hombre ?sobrino)
101 =>
102 (assert (sobrino-de ?sobrino ?tios))
103 (printout t ?sobrino " es sobrino de " ?tios crlf)
104 )
105 (defrule sobrina
106 (tios ?tios ?sobrina)
107 (mujer ?sobrina)
108 =>
109 (assert (sobrina-de ?sobrina ?tios))
110 (printout t ?sobrina " es sobrina de " ?tios crlf )
111 )
112 ; -----Bisabuelos-----
113 (defrule bisabuelos
114 (progenitor-de ?abuelo ?padre)
115 (bisabuelo-de ?abuelo ?nieto)
116 =>
117 (assert (bisabuelos-de ?abuelo ?nieto))
118 )
119 (defrule bisabuelo
120 (bisabuelo-de ?abuelo ?nieto)
121 (hombre ?abuelo)
```

```
122 =>
123 (assert (bisabuelo-de ?abuelo ?nieto))
124 (printout t ?abuelo " es bisabuelo de " ?nieto crlf)
125 )
126 (defrule bisabuela
127 (bisabuela-de ?abuela ?nieto)
128 (mujer ?abuela)
129 =>
130 (assert (bisabuela-de ?abuela ?nieto))
131 (printout t ?abuela " es bisabuela de " ?nieto crlf)
132 )
133 ;-----Primos-----
134 (defrule primos
135 (primos ?primol ?primo2)
136 (sobrino-de ?tio1 ?primo2)
137 (sobrino-de ?tio2 ?primol)
138 (test (neq ?primol ?primo2))
139 =>
140 (assert (primos ?primol ?primo2))
141 )
142 (defrule primo
143 (sobrino-de ?tio1 ?primo2)
144 (sobrino-de ?tio2 ?primol)
145 (hombre ?primol)
146 =>
147 (assert (primo-de ?primol ?primo2))
148 (printout t ?primol " primo de " ?primo2 crlf)
149 )
150 (defrule prima
151 (sobrina-de ?tio1 ?primo2)
152 (sobrina-de ?tio2 ?primol)
153 (mujer ?primol)
154 =>
155 (assert (prima-de ?primol ?primo2))
156 (printout t ?primol " prima de " ?primo2 crlf)
157 )
158 (defrule tio
159 (hermanos ?hijo1 ?hijo2)
160 (hombre ?hijo2)
161 (progenitor-de ?hijo1 ?hijo)
162 =>
163 (assert (tio-de ?hijo2 ?hijo))
164 (printout t ?hijo2 " es tio de " ?hijo crlf)
165 )
166 (defrule tia
167 (hermanos ?hijo1 ?hijo2)
168 (mujer ?hijo2)
169 (progenitor-de ?hijo1 ?hijo)
170 =>
171 (assert (tia-de ?hijo2 ?hijo))
172 (printout t ?hijo2 " es tia de " ?hijo crlf)
173 )
174 (defrule sobrino
175 (hermanos ?hijo1 ?hijo2)
176 (progenitor-de ?hijo1 ?hijo)
177 (hombre ?hijo)
178 =>
179 (assert (sobrino-de ?hijo ?hijo2))
180 (printout t ?hijo " es sobrino de " ?hijo2 crlf)
181 )
182 (defrule sobrina
```

```
183 (hermanos ?hijo1 ?hijo2)
184 (progenitor-de ?hijo1 ?hijo)
185 (mujer ?hijo)
186 =>
187 (assert (sobrina-de ?hijo ?hijo2))
188 (printout t ?hijo " es sobrina de " ?hijo2 crlf)
189 )
190 (defrule primo
191 (hermanos ?hijo1 ?hijo2)
192 (progenitor-de ?hijo1 ?hijo3)
193 (progenitor-de ?hijo2 ?hijo4)
194 (hombre ?hijo3)
195 =>
196 (assert (primo-de ?hijo3 ?hijo4))
197 (printout t ?hijo3 " es primo de " ?hijo4 crlf)
198 )
199 (defrule prima
200 (hermanos ?hijo1 ?hijo2)
201 (progenitor-de ?hijo1 ?hijo3)
202 (progenitor-de ?hijo2 ?hijo4)
203 (mujer ?hijo3)
204 =>
205 (assert (prima-de ?hijo3 ?hijo4))
206 (printout t ?hijo3 " es prima de " ?hijo4 crlf)
207 )
208
209 ; --- Hechos ----
210 (deffacts inicio
211 (hombre Fabian)
212 (hombre Ricardo)
213 (mujer Fabiola)
214 (hombre Andres)
215 (mujer Rosa)
216 (mujer Maria)
217 (hombre Mario)
218 (mujer Irene)
219 (hombre David)
220 (hombre Alejandro)
221 (mujer Andrea)
222 (hombre Luis)
223 (hombre Enrique)
224 (bisabuelo-de Fabian Luis)
225 (progenitor-de Ricardo David)
226 (progenitor-de Ricardo Alejandro)
227 (progenitor-de Ricardo Mario)
228 (sobrina-de Andrea Mario)
229 (marido-de Ricardo Fabiola)
230 (progenitor-de David Luis)
231 (marido-de David Rosa)
232 (hermano-de Alejandro David)
233 (tio-de Alejandro Luis)
234 )
```

In [ ]:

1

