10/02/14 02:51:57 /home-reseau/tchapon/4INFO/Prolog/tp4/tp4_arbres_etud.pl

```
  1  /*["~/4INFO/Prolog/tp4/tp4_arbres_etud.pl"].*/
  2  /**
  3  TP 4 Arbres binaires - Prolog
  4
  5  @author Theo CHAPON
  6  @author Hassan El OMARI ALAOUI
  7  @version Annee scolaire 2014/2015
  8  */
  9
 10
 11  /*
 12  -------------------------------------------------------------------------------
 13  Définition des prédicats
 14  -------------------------------------------------------------------------------
 15  */
 16
 17  /*
 18  Question 1 : B est un arbre binaire d'entier
 19  */
 20  arbre_binaire(arb_bin(R,vide,vide)):-integer(R).
 21  arbre_binaire(arb_bin(R,G,vide)):-integer(R),arbre_binaire(G).
 22  arbre_binaire(arb_bin(R,vide,D)):-integer(R),arbre_binaire(D).
 23  arbre_binaire(arb_bin(R,G,D)):- integer(R),arbre_binaire(G),arbre_binaire(D).
 24  /*
 25  arbre_binaire(arb_bin(1, arb_bin(2, arb_bin(6, vide, vide), vide), arb_bin(3, arb_bin(4, vide, vide),
     arb_bin(5, vide, vide)))).
 26      Yes (0.00s cpu, solution 1, maybe more)
 27
 28  arbre_binaire(arb_bin(1, arb_bin(2, arb_bin(6, vide, vide), vide), arb_bin(3, arb_bin(4, vide, vide),
     arb_bin("5", vide, vide)))).
 29      No (0.00s cpu)
 30
 31  */
 32
 33  /*
 34  Question 2 : E est l'une des étiquettes de B
 35  */
 36  dans_arbre_binaire(E,arb_bin(E,_,_)):-!.
 37  dans_arbre_binaire(E,arb_bin(_,G,_)):-dans_arbre_binaire(E,G),!.
 38  dans_arbre_binaire(E,arb_bin(_,_,D)):-dans_arbre_binaire(E,D),!.
 39  /*
 40  On met un cut car si on trouve que E est égal à l'une des racines de l'arbre binaire, il n'est pas
     nécessaire de boucler sur les autres branches.
 41
 42  dans_arbre_binaire(3,arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, vide, vide))).
 43      Yes (0.00s cpu, solution 1, maybe more)
 44  dans_arbre_binaire(6,arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, vide, vide))).
 45      No (0.00s cpu)
 46  */
 47
 48
 49  /*
 50  Question 3 : S est un sous arbre de B
 51  */
 52  sous_arbre_binaire(S,S):-!.
 53  sous_arbre_binaire(S,arb_bin(_,G,_)):-sous_arbre_binaire(S,G),!.
 54  sous_arbre_binaire(S,arb_bin(_,_,D)):-sous_arbre_binaire(S,D),!.
 55  /*
 56  On met un cut car si on trouve le sous arbre S de l'arbre binaure B, il n'est pas nécessaire de boucler
     sur les autres branches.
 57
 58  sous_arbre_binaire(arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, 7, vide)),arb_bin(3, arb_bin(4, vide,
     vide), arb_bin(5, 7, vide))).
 59      Yes (0.00s cpu)
 60
 61  sous_arbre_binaire(arb_bin(5,8,vide),arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, 7, vide))).
 62      No (0.00s cpu)
 63
 64  sous_arbre_binaire(arb_bin(5,7,vide),arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, 7, vide))).
 65      Yes (0.00s cpu)
 66  */
 67
```

```
 68  /*
 69  Question 4 : B1 est l'arbre B où toute occurrence du sous arbre SA1 est remplacée par le sous arbre SA2
 70  */
 71  remplacer(_,_,arb_bin(R,vide,vide),arb_bin(R,vide,vide)).
 72  remplacer(SA1,SA2,arb_bin(R,vide,SA1),arb_bin(R,vide,SA2)).
 73  remplacer(SA1,SA2,arb_bin(R,SA1,vide),arb_bin(R,SA2,vide)).
 74  remplacer(SA1,SA2,arb_bin(R,G,vide),arb_bin(R,B,vide)):-remplacer(SA1,SA2,G,B).
 75  remplacer(SA1,SA2,arb_bin(R,vide,D),arb_bin(R,vide,B)):-remplacer(SA1,SA2,D,B).
 76
 77  remplacer(SA1,SA2,arb_bin(R,SA1,SA1),arb_bin(R,SA2,SA2)).
 78  remplacer(SA1,SA2,arb_bin(R,G,SA1),arb_bin(R,B,SA2)):- \==(G,SA1),remplacer(SA1,SA2,G,B).
 79  remplacer(SA1,SA2,arb_bin(R,SA1,D),arb_bin(R,SA2,B)):- \==(D,SA1),remplacer(SA1,SA2,D,B).
 80  remplacer(SA1,SA2,arb_bin(R,G,D),arb_bin(R,G1,D1)):- \==(SA1,G),
     \==(SA1,D),remplacer(SA1,SA2,G,G1),remplacer(SA1,SA2,D,D1).
 81  /*
 82  remplacer(arb_bin(3,vide,vide),arb_bin(4,vide,vide),arb_bin(1,arb_bin(3,vide,vide),arb_bin(3,vide,vide)),
     B).
 83      B = arb_bin(1, arb_bin(4, vide, vide), arb_bin(4, vide, vide))
 84      Yes (0.00s cpu, solution 1, maybe more)
 85
 86  remplacer(arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, vide, vide)),arb_bin(3, arb_bin(4, vide, vide),
     arb_bin(5, 7, vide)),arb_bin(1, arb_bin(2, arb_bin(6, vide, vide), vide), arb_bin(3, arb_bin(4, vide,
     vide), arb_bin(5, vide, vide))),B).
 87      B = arb_bin(1, arb_bin(2, arb_bin(6, vide, vide), vide), arb_bin(3, arb_bin(4, vide, vide),
     arb_bin(5, 7, vide)))
 88      Yes (0.00s cpu, solution 1, maybe more)
 89
 90  remplacer(arb_bin(3,vide,vide),arb_bin(4,vide,vide),arb_bin(1,arb_bin(3,vide,vide),arb_bin(5,arb_bin(3,vi
     de,vide),arb_bin(9,vide,vide))),B).
 91      B = arb_bin(1, arb_bin(4, vide, vide), arb_bin(5, arb_bin(4, vide, vide), arb_bin(9, vide, vide)))
 92      Yes (0.00s cpu, solution 1, maybe more)
 93
 94  remplacer(arb_bin(3,vide,vide),arb_bin(4,vide,vide),arb_bin(1,arb_bin(6,vide,vide),arb_bin(5,arb_bin(8,vi
     de,vide),arb_bin(9,vide,vide))),B).
 95      B = arb_bin(1, arb_bin(6, vide, vide), arb_bin(5, arb_bin(8, vide, vide), arb_bin(9, vide, vide)))
 96      Yes (0.00s cpu, solution 1, maybe more)
 97  */
 98
 99  /*
100      Question 5 : B1 et B2 sont isomorphes
101  */
102  isomorphes(arb_bin(R,vide,vide),arb_bin(R,vide,vide)).
103  isomorphes(arb_bin(R,G1,D1),arb_bin(R,G2,D2)):- isomorphes(G1,G2),isomorphes(D1,D2).
104  isomorphes(arb_bin(R,G1,D1),arb_bin(R,G2,D2)):- isomorphes(G1,D2),isomorphes(D1,G2).
105  /*
106  isomorphes(arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, arb_bin(6, vide, vide), arb_bin(7, vide,
     vide))),arb_bin(3, arb_bin(5, arb_bin(6, vide, vide), arb_bin(7, vide, vide)), arb_bin(4, vide, vide))).
107      Yes (0.00s cpu, solution 1, maybe more)
108  */
109
110  /*
111      Question 6 : L contient les informations de l'arbre B en le parcourant en infixe
112  */
113  /***Fonction auxiliaire****/
114  concat([A|X],Z,[A|T]):- concat(X,Z,T).
115  concat([],Z,Z).
116  /*************************/
117
118  infixe(vide,[]).
119  infixe(arb_bin(R,G,D),L) :- infixe(G,L1), infixe(D,L2), concat(L1,[R|L2],L).
120  /*
121  infixe(arb_bin(1, arb_bin(2, arb_bin(6, vide, vide), vide), arb_bin(3, arb_bin(4, vide, vide), arb_bin(5,
     vide, vide))),L).
122      L = [6, 2, 1, 4, 3, 5]
123      Yes (0.00s cpu)
124  */
125
126  /*
127      Question 7 : B2 est l’arbre ordonné d’entiers obtenu par l’insertion de la valeur X dans l’arbre
     ordonné d’entiers B1.
128  */
129  insertion_arbre_ordonne(X,vide,arb_bin(X,vide,vide)).
130  insertion_arbre_ordonne(X,arb_bin(X,G,D),arb_bin(X,G,D)).
131  insertion_arbre_ordonne(X,arb_bin(R,G,D1),arb_bin(R,G,D2)) :-
132      >(X,R),
```

```
133      insertion_arbre_ordonne(X,D1,D2).
134  insertion_arbre_ordonne(X,arb_bin(R,G1,D),arb_bin(R,G2,D)) :-
135      <(X,R),
136      insertion_arbre_ordonne(X,G1,G2).
137  /*
138  insertion_arbre_ordonne(1,arb_bin(8, arb_bin(4, arb_bin(2, vide, vide), arb_bin(6, vide, vide)),
     arb_bin(12, arb_bin(10, vide, vide), vide)),B).
139      B = arb_bin(8, arb_bin(4, arb_bin(2, arb_bin(1, vide, vide), vide), arb_bin(6, vide, vide)),
     arb_bin(12, arb_bin(10, vide, vide), vide))
140      Yes (0.00s cpu)
141
142  insertion_arbre_ordonne(12,arb_bin(8, arb_bin(4, arb_bin(2, vide, vide), arb_bin(6, vide, vide)),
     arb_bin(12, arb_bin(10, vide, vide), vide)),B).
143      B = arb_bin(8, arb_bin(4, arb_bin(2, vide, vide), arb_bin(6, vide, vide)), arb_bin(12, arb_bin(10,
     vide, vide), vide))
144      Yes (0.00s cpu, solution 1, maybe more)
145  */
146
147  /*
148      Question 8 :
149  */
150  insertion_arbre_ordonne1(X,arb_bin(_,A,_)):-free(A),A = arb_bin(X,_,_),!.
151  insertion_arbre_ordonne1(X,arb_bin(X,_,_)).
152  insertion_arbre_ordonne1(X,arb_bin(R,_,D)) :-
153      >(X,R),insertion_arbre_ordonne1(X,D).
154  insertion_arbre_ordonne1(X,arb_bin(R,G,_)) :-
155      <(X,R),insertion_arbre_ordonne1(X,G).
156
157  /*
158  insertion_arbre_ordonne2(1,arb_bin(8, arb_bin(4, arb_bin(2, _, _), arb_bin(6, _, _)), arb_bin(12,
     arb_bin(10, _, _), _)),B).
159      B = arb_bin(8, arb_bin(4, arb_bin(2, arb_bin(1, _233, _234), _84), arb_bin(6, _95, _96)), arb_bin(12,
     arb_bin(10, _115, _116), _122))
160      Yes (0.00s cpu)
161
162  insertion_arbre_ordonne2(4,arb_bin(8, arb_bin(4, arb_bin(2, _, _), arb_bin(6, _, _)), arb_bin(12,
     arb_bin(10, _, _), _)),B).
163      B = arb_bin(8, arb_bin(4, arb_bin(2, _83, _84), arb_bin(6, _95, _96)), arb_bin(12, arb_bin(10, _115,
     _116), _122))
164      Yes (0.00s cpu, solution 1, maybe more)
165  */
166  /*
167  ------------------------------------------------------------------------------
168  Tests
169  ------------------------------------------------------------------------------
170  */
171
172
173  % Quelques arbres à copier coller pour vous faire gagner du temps, mais
174  % n'hésitez pas à en définir d'autres
175
176  /*
177  arb_bin(1, arb_bin(2, arb_bin(6, vide, vide), vide), arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, vide,
     vide)))
178
179  arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, vide, vide))
180
181  arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, 7, vide))
182
183  arb_bin(3, arb_bin(4, vide, vide), arb_bin(5, arb_bin(6, vide, vide), arb_bin(7, vide, vide)))
184
185  arb_bin(3, arb_bin(5, arb_bin(6, vide, vide), arb_bin(7, vide, vide)), arb_bin(4, vide, vide))
186
187  arb_bin(3, arb_bin(6, vide, vide), arb_bin(5, arb_bin(4, vide, vide), arb_bin(7, vide, vide)))
188
189  arb_bin(8, arb_bin(4, arb_bin(2, vide, vide), arb_bin(6, vide, vide)), arb_bin(12, arb_bin(10, vide,
     vide), vide))
190
191  arb_bin(8, arb_bin(4, arb_bin(2, _, _), arb_bin(6, _, _)), arb_bin(12, arb_bin(10, _, _), _))
192
193  arb_bin(6,arb_bin(2,arb_bin(1,vide,vide),arb_bin(4,vide,vide)),arb_bin(8,vide,arb_bin(10,vide,vide)))
194
195  arb_bin(8,arb_bin(2,arb_bin(1,vide,vide),arb_bin(4,vide,vide)),arb_bin(6,vide,arb_bin(10,vide,vide)))
196
```

```
197  arb_bin(6,arb_bin(2,arb_bin(1,vide,vide),arb_bin(4,vide,vide)),arb_bin(8,arb_bin(2,arb_bin(1,vide,vide),a
     rb_bin(4,vide,vide)),arb_bin(10,vide,vide)))
198
199  */
```