with arbre\_binaire;

procedure test\_arbre\_binaire is

    package ABR\_Char is

        new Arbre\_Binaire (T\_Value => Character, Zero => '0');

    use ABR\_Char;

    A: T\_Branch;

    procedure Exemple\_Arbre(*A*:in out T\_Branch) is

        begin

            Initialiser(20,A);

            Inserer(22,'A',A);

            Inserer(23,'B',A);

            Inserer(16,'C',A);

            Inserer(10,'D',A);

            Inserer(21,'E',A);

        end Exemple\_Arbre;

    procedure Tester\_Exemple\_Arbre is

        begin

            Exemple\_Arbre(A);

            --Tester l'existence et la bonne répartition des clés au sein de l'arbre.

                --En utilisant les fonctions NodeKey, NodeValue, Fils\_Gauche, Fils\_Droit et Est\_Nul;

            pragma assert(not Est\_Nul(A));                                  --S'assurer que A n'est pas vide.

            pragma assert(NodeKey(A)=20);                                   --S'assurer que la clé de la racine est 20.

            pragma assert(not Est\_Nul(Fils\_Gauche(A)));                     --S'assurer que le fils gauche n'est pas vide.

            pragma assert(NodeKey(Fils\_Gauche(A))=16);                      --S'assurer que sa clé est 16.

            pragma assert(NodeValue(Fils\_Gauche(A))='C');                   --S'assurer que sa donnée est 'C'.

            pragma assert(not Est\_Nul(Fils\_Gauche(Fils\_Gauche(A))));        --S'assurer que le fils gauche de 16 n'est pas vide.

            pragma assert(Est\_Nul(Fils\_Droit(Fils\_Gauche(A))));             --S'assurer que le fils droit de 16 est vide.

            pragma assert(NodeKey(Fils\_Gauche(Fils\_Gauche(A)))=10);         --S'assurer que la clé du fils gauche de 16 est 10.

            pragma assert(NodeValue(Fils\_Gauche(Fils\_Gauche(A)))='D');      --S'assurer que la donnée du fils gauche de 16 est 'D'.

            pragma assert(not Est\_Nul(Fils\_Droit(A)));                      --S'assurer que le fils droit n'est pas vide.

            pragma assert(NodeKey(Fils\_Droit(A))=22);                       --S'assurer que sa clé est 22.

            pragma assert(NodeValue(Fils\_Droit(A))='A');                    --S'assurer que sa donnée est 'A'.

            pragma assert(not Est\_Nul(Fils\_Droit(Fils\_Droit(A))));          --S'assurer que son fils droit n'est pas vide.

            pragma assert(NodeKey(Fils\_Droit(Fils\_Droit(A)))=23);           --S'assurer que la clé du fils droit de 22 est 23.

            pragma assert(NodeValue(Fils\_Droit(Fils\_Droit(A)))='B');        --S'assurer que la donnée du fils droit de 22 est 'B'.

            pragma assert(not Est\_Nul(Fils\_Gauche(Fils\_Droit(A))));         --S'assurer que le fils gauche de 22 n'est pas vide.

            pragma assert(NodeKey(Fils\_Gauche(Fils\_Droit(A)))=21);          --S'assurer que la clé du fils gauche de 22 est 21.

            pragma assert(NodeValue(Fils\_Gauche(Fils\_Droit(A)))='E');       --S'assurer que la donnée du fils gauche de 22 est 'E'.

                --En utilisant la fonction Rech\_Noeud.

            pragma assert(not Est\_Nul(Rech\_Noeud(21,A)));   --S'assurer que la clé 21 existe dans l'arbre.

            pragma assert(not Est\_Nul(Rech\_Noeud(10,A)));   --S'assurer que la clé 10 existe dans l'arbre.

            pragma assert(Est\_Nul(Rech\_Noeud(69,A)));       --S'assurer que la clé 69 n'existe pas dans l'arbre.

                --En utilisant la fonction Rech\_Ancetre.

            pragma assert(NodeKey(Rech\_Ancetre(16,A))=20);  --S'assurer que l'ancêtre de 16 est 20.

            pragma assert(NodeKey(Rech\_Ancetre(10,A))=16);  --S'assurer que l'ancêtre de 10 est 16.

            pragma assert(NodeKey(Rech\_Ancetre(22,A))=20);  --S'assurer que l'ancêtre de 22 est 20.

            pragma assert(NodeKey(Rech\_Ancetre(21,A))=22);  --S'assurer que l'ancêtre de 21 est 22.

            pragma assert(NodeKey(Rech\_Ancetre(23,A))=22);  --S'assurer que l'ancêtre de 23 est 22.

                --En utilisant la fonction Gen.

            pragma assert(Gen(21,A)=2);                     --S'assurer que la génération de 21 est 2.

            pragma assert(Gen(23,A)=2);                     --S'assurer que la génération de 23 est 2.

            pragma assert(Gen(10,A)=2);                     --S'assurer que la génération de 10 est 2.

            pragma assert(Gen(16,A)=1);                     --S'assurer que la génération de 16 est 1.

            pragma assert(Gen(22,A)=1);                     --S'assurer que la génération de 22 est 1.

            pragma assert(Gen(20,A)=0);                     --S'assurer que la génération de 2. est 0.

                --En utilisant la fonction Nbr\_Fils\_Noeud.

            pragma assert(Nbr\_Fils\_Noeud(20,A)=5);          --S'assurer que le nombre de fils de 20 est 5.

            pragma assert(Nbr\_Fils\_Noeud(16,A)=1);          --S'assurer que le nombre de fils de 16 est 1.

            pragma assert(Nbr\_Fils\_Noeud(10,A)=0);          --S'assurer que le nombre de fils de 10 est 0.

            pragma assert(Nbr\_Fils\_Noeud(22,A)=2);          --S'assurer que le nombre de fils de 22 est 2.

            pragma assert(Nbr\_Fils\_Noeud(21,A)=0);          --S'assurer que le nombre de fils de 21 est 0.

            pragma assert(Nbr\_Fils\_Noeud(23,A)=0);          --S'assurer que le nombre de fils de 23 est 0.

                --En utilisant Nbr\_Meme\_Generation.

            pragma assert(Nbr\_Meme\_Generation(0,A)=1);      --S'assurer que le nombre de clés de génération 0 est 1.

            pragma assert(Nbr\_Meme\_Generation(1,A)=2);      --S'assurer que le nombre de clés de génération 1 est 2.

            pragma assert(Nbr\_Meme\_Generation(2,A)=3);      --S'assurer que le nombre de clés de génération 2 est 3.

        end Tester\_Exemple\_Arbre;

    procedure Tester\_Supprimer is

        begin

            Detruire(A);

            pragma assert(Est\_Nul(A));                      --S'assurer que A a été détruit.

            Exemple\_Arbre(A);

            Supprimer\_Cle\_ET\_Fils(10,A);

            pragma assert(Est\_Nul(Rech\_Noeud(10,A)));       --S'assurer que le noeud de clé 10 a été supprimé.

            Supprimer\_Cle\_ET\_Fils(22,A);

            pragma assert(Est\_Nul(Fils\_Droit(A)));          --S'assurer que le fils droit et ses fils ont été supprimés.

            Supprimer\_Cle\_ET\_Fils(20,A);

            pragma assert(Est\_Nul(A));                      --S'assurer que A a été détruit.

        end Tester\_Supprimer;

    begin

        Tester\_Exemple\_Arbre;

        Tester\_Supprimer;

    end test\_arbre\_binaire;