```
with Arbre_Genealogique;            use Arbre_Genealogique;



procedure test_arbre_genealogique is


    AG:Arbre_Genealogique.Arbre_Binaire_Character.T_Branch;
    procedure Exemple_Arbre(AG:in out Arbre_Genealogique.Arbre_Binaire_Charact
er.T_Branch) is
        begin
            Init_AG(18,AG);
            Ajouter_Ancetre(2,'P',18,AG);
            Ajouter_Ancetre(19,'M',18,AG);
            Ajouter_Ancetre(1,'P',2,AG);
            Ajouter_Ancetre(15,'M',2,AG);
            Ajouter_Ancetre(4,'P',15,AG);
            Ajouter_Ancetre(33,'M',19,AG);
            Ajouter_Ancetre(25,'P',33,AG);
            Ajouter_Ancetre(42,'M',33,AG);
            Ajouter_Ancetre(35,'P',42,AG);
        end Exemple_Arbre;

    procedure Tester_Exemple_Arbre is
        begin
            Exemple_Arbre(AG);
            --Tester la création et bonne répartition des ancêtres dans l'AG.
            pragma assert(not Est_Nul_AG(AG));
            pragma assert(Est_Present(2,AG));
            pragma assert(Est_Present(19,AG));
            pragma assert(Est_Present(1,AG));
            pragma assert(Est_Present(15,AG));
            pragma assert(Est_Present(4,AG));
            pragma assert(Est_Present(33,AG));
            pragma assert(Est_Present(25,AG));
            pragma assert(Est_Present(42,AG));
            pragma assert(Est_Present(35,AG));
            pragma assert(not Est_Present(64,AG));

            --
Tester le nombre d'ancêtres d'un noeud, ce dernier est inclu dans le calcul.
            pragma assert(Nombre_Ancetres(2,AG)=4);
            pragma assert(Nombre_Ancetres(19,AG)=5);
            pragma assert(Nombre_Ancetres(18,AG)=10);
            pragma assert(Nombre_Ancetres(33,AG)=4);
            pragma assert(Nombre_Ancetres(15,AG)=2);
            pragma assert(Nombre_Ancetres(4,AG)=1);
            pragma assert(Nombre_Ancetres(25,AG)=1);
       end Tester_Exemple_Arbre;
```

```ada
procedure Tester_Supprimer is
     begin
         Exemple_Arbre(AG);
         --Tester la suppression
         Supprimer_famille(35,AG);
         pragma assert(not Est_Present(35,AG));
         Supprimer_famille(25,AG);
         pragma assert(not Est_Present(25,AG));
         Supprimer_famille(4,AG);
         pragma assert(not Est_Present(4,AG));
         Supprimer_famille(15,AG);
         pragma assert(not Est_Present(15,AG));
         Supprimer_famille(1,AG);
         pragma assert(not Est_Present(1,AG));
         Supprimer_famille(19,AG);
         pragma assert(not Est_Present(19,AG));
         pragma assert(not Est_Present(33,AG));
         pragma assert(not Est_Present(42,AG));
         Supprimer_famille(18,AG);
         pragma assert(Est_Nul_AG(AG));
     end Tester_Supprimer;
 begin
     Tester_Exemple_Arbre;
     Tester_Supprimer;
 end test_arbre_genealogique;
```