```
with Ada.Integer_Text_IO;           use Ada.Integer_Text_IO;

package body Arbre_Genealogique is

    procedure Afficher_Ensemble2 is new Arbre_Binaire_Character.Piles_Cle.Affi
cher_Pile(Afficher_Element => Afficher_Entier);
                                    --TESTS--

    procedure Init_AG(Cle: in Integer; AG: out T_Branch) is
        begin
            Initialiser(Cle,AG);
        end Init_AG;

    function Est_Nul_AG(AG:in T_Branch) return Boolean is
        begin
            return Est_Nul(AG);
        end Est_Nul_AG;

    function Est_Present(Cle: in Integer; AG: in T_Branch) return Boolean is
        begin
            if not Est_Nul(AG) then
                return (not Est_Nul(Rech_Noeud(Cle,AG)));
            else
                return False;
            end if;
        end Est_Present;

    procedure Init_Foret(F_Foret: in out Foret.T_Pile) is
        choix:Character;
        Cle:Integer;
        AG:T_Branch;
        begin
            Foret.Initialiser(F_Foret); --Initialisation de la pile de forêt
            Put("Saisissez la clé par laquelle vous voulez initialiser le prem
ier arbre de la forêt : "); Get(Cle);New_Line;
            while Cle=-181199 or Cle=-34404 loop
                Put("Les valeurs -181199 et -
34404 sont utilisées intérieurement pour assurer le fonctionnement de l'applic
ation. Veuillez saisir une autre valeur : "); Get(Cle);New_Line;
            end loop;
            loop
                Put("Voulez-
vous ajouter un autre arbre généalogique à la forêt? [y/n] : ");Get(choix);New
_Line;
                    if choix/='n' and choix/='N' then
                        Put("Saisissez la clé par laquelle vous voulez initial
iser cet arbre : "); Get(Cle);New_Line;
                        while Cle=-181199 or Cle=-34404 loop
```

```
                            Put("Les valeurs -181199 et -
34404 sont utilisées intérieurement pour assurer le fonctionnement de l'applic
ation. Veuillez saisir une autre valeur : "); Get(Cle);New_Line;
                        end loop;
                        New_Line;
                        Init_AG(Cle,AG); --
Initialisation de l'arbre généalogique par la clé Cle
                        Foret.Empiler(F_Foret, AG); --Empiler l'AG
                    end if;
                exit when choix='n' or choix ='N'; --
Le programme s'arrête quand l'utilisateur a ajouté autant d'AG qu'il veuille.
            end loop;
        end Init_Foret;


                        --FONCTIONS/PROCEDURES GENERATIONNELLES-

    function Nombre_Ancetres(Descendant: in Integer; AG: in T_Branch) return I
nteger is
        begin
            return 1 + Nbr_Fils_Noeud(Descendant,AG);
        end Nombre_Ancetres;

    procedure Ensemble_Ancetres_Noeud(Descendant: in Integer; AG: in T_Branch)
 is
        Ens:Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if Est_Nul_AG(AG) then
                Put_Line("Impossible de chercher. L'arbre est vide!");
            else
                Ens:=Ensemble_Fils_Noeud(Descendant,AG);New_Line;
                if Arbre_Binaire_Character.Piles_Cle.Est_Vide(Ens) then
                    Put_Line("Cet individu n'a aucun parent connu.");New_Line;
                else
                    Put_Line("Les ancêtres de la clé" & Integer'Image(Descenda
nt) & " sont : ");
                    Afficher_Ensemble2(Ens);New_Line;New_Line;
                end if;
            end if;
        end Ensemble_Ancetres_Noeud;

    procedure Ensemble_Ancetres_Meme_Generation(g,Descendant:in Integer; AG: i
n T_Branch) is
        begin
            if g<0 then
                Put_Line("La génération saisie est invalide!");
            elsif g=0 then
                Put_Line("L'ancêtre de génération 0 par rapport au descendant"
 & Integer'Image(Descendant) & " est lui-même!");
```

```
            else
                Put("Les ancêtres de génération" & Integer'Image(g) & " par ra
pport à" & Integer'Image(Descendant) & " sont: ");
                Ensemble_Meme_Generation(g,Descendant,AG); New_Line;
            end if;
        end Ensemble_Ancetres_Meme_Generation;

    procedure Ensemble_Ancetres_Generation_N(g,Descendant: in Integer; AG: in
T_Branch) is
        begin
            if g<0 then
                Put_Line("La génération saisie est invalide!");
            elsif g=0 then
                Put_Line("L'ancêtre de génération 0 par rapport au descendant"
 & Integer'Image(Descendant) & " est lui-même!");
            else
                Put_Line("Les ancêtres de génération" & Integer'Image(g) & " o
u moins par rapport au descendant" & Integer'Image(Descendant) & " sont:");
                Ensemble_n_Generation(g,Descendant,AG);New_Line;
            end if;
        end Ensemble_Ancetres_Generation_N;

    procedure Ensemble_Un_Parent(AG: in T_Branch) is
        Ens: Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if not Est_Nul(AG) then
                Ens:=Ensemble_Un_Fils(AG);New_Line;
                if Arbre_Binaire_Character.Piles_Cle.Est_Vide(Ens) then
                    Put_Line("Il n'y a aucun individu dont un seul parent est
connu.");New_Line;
                else
                    Put("Les individus ayant un seul parent connu sont:");
                    Afficher_Ensemble2(Ens);New_Line;New_Line;
                end if;
            else
                Put_Line("Impossible de chercher. L'arbre est vide.");
            end if;
        end Ensemble_Un_Parent;

    procedure Ensemble_Deux_Parents(AG: in T_Branch) is
        Ens: Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if not Est_Nul(AG) then
                Ens:=Ensemble_Deux_Fils(AG);
                if Arbre_Binaire_Character.Piles_Cle.Est_Vide(Ens) then
                    Put_Line("Il n'y a aucun individu dont les deux parents so
nt connus.");New_Line;
                else
```

```
                    Put("Les individus ayant les deux parents connus sont:");
                    Afficher_Ensemble2(Ens);New_Line;New_Line;
                end if;
            else
                Put_Line("Impossible de chercher. L'arbre est vide.");
            end if;
        end Ensemble_Deux_Parents;

    procedure Ensemble_Orphelins(AG: in T_Branch) is
        Ens: Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if not Est_Nul(AG) then
                Ens:=Ensemble_Feuilles(AG);
                Put("Les individus n'ayant aucun parent connu sont:");
                Afficher_Ensemble2(Ens);New_Line;New_Line;
            else
                Put_Line("Impossible de chercher. L'arbre est vide.");
            end if;
        end Ensemble_Orphelins;


                        --FONCTIONS/PROCEDURES DE RECHERCHE--

    function Rech_Noeud_AG(Cle: in Integer; AG: in T_Branch) return T_Branch i
s
        begin
            return Rech_Noeud(Cle,AG);
        end Rech_Noeud_AG;

    procedure Affecter_Rech_Noeud_AG(Cle: in Integer;AG: in T_Branch; Noeud: i
n out T_Branch) is
        begin
            Affecter_Rech_Noeud(Cle,AG,Noeud);
        end Affecter_Rech_Noeud_AG;

    function Access_Tree_Forest(l:in Integer; F_Foret: in Foret.T_Pile) return
 T_Branch is
        Foret_V:Foret.T_Pile;
        begin
            if l=Foret.Size_Pile(F_Foret) then
                return Sommet(F_Foret);
            else
                Affecter_Pile(Foret_V,F_Foret);
                for i in 1..(Size_Pile(F_Foret)-l) loop
                    Affecter_Pile(Foret_V,Next_Pile(Foret_V));
                end loop;
                return Sommet(Foret_V);
            end if;
```

```
        end Access_Tree_Forest;

    function Descendants_Noeud_Foret(Cle: in Integer; F_Foret: in Foret.T_Pile
) return Arbre_Binaire_Character.Piles_Cle.T_Pile is
        Parcours: Foret.T_Pile;
        Recherche: T_Branch;
        Cle_Desc:Integer;
        Ens_Noeud: Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if Cle=-34404 or Foret.Est_Vide(F_Foret) then --
Si l'exception Arbre_Vide a été soulevé dans Half_Sibling_Foret ou si la forêt
 est vide
                Arbre_Binaire_Character.Piles_Cle.Initialiser(Ens_Noeud);
                return Ens_Noeud; --retourner un ensemble vide
            else
                Initialiser(Parcours);
                Affecter_Pile(Parcours,F_Foret); --
Notre parcours dans la forêt
                Arbre_Binaire_Character.Piles_Cle.Initialiser(Ens_Noeud); --
L'éventuel ensemble de descendants de la clé Cle
                while (not Foret.Est_Vide(Parcours)) loop --
Tant que la pile d'arbres n'est pas vide
                    Affecter_Rech_Noeud_AG(Cle,Foret.Sommet(Parcours),Recherch
e); --
Recherche reçoit le noeud contenant la clé Cle dans l'arbre sommet de la pile
F_Foret
                    if not Est_Nul_AG(Recherche) then --
Si on a trouvé un noeud de clé Cle dans cet arbre
                        Cle_Desc:=Nodekey(Rech_Ancetre(Nodekey(Recherche),Fore
t.Sommet(Parcours))); --Cle_Desc reçoit le descendant de cette clé
                        if Cle_Desc/=-34404 then --
Si aucune erreur n'a été soulevée
                            Arbre_Binaire_Character.Piles_Cle.Empiler(Ens_Noeu
d,Cle_Desc); --On ajoute ce descendant dans l'ensemble
                        end if;
                    end if;
                    Affecter_Pile(Parcours,Next_Pile(Parcours)); --
On dépile Parcours pour parcourir tous les arbres de la forêt
                end loop;
                return Ens_Noeud; --
Retourner l'ensemble des clés des descendants de l'individu Cle
            end if;
        end Descendants_Noeud_Foret;

    procedure Afficher_Descendants_Noeud_Foret(Cle: in Integer; F_Foret: in Fo
ret.T_Pile) is
        begin
```

```
            Afficher_Ensemble2(Descendants_Noeud_Foret(Cle, F_Foret)); --
Affichage des clés
                    New_Line;
        end Afficher_Descendants_Noeud_Foret;

    procedure Half_Sibling_Foret(Cle: in Integer; AG: in T_Branch; F_Foret: in
 Foret.T_Pile) is
        function Half_Siblings(Cle: in Integer; AG: in T_Branch; F_Foret: in F
oret.T_Pile) return Arbre_Binaire_Character.Piles_Cle.T_Pile is
            Noeud_Cle: constant T_Branch:=Rech_Noeud_AG(Cle,AG);
            Desc_Pere,Desc_Mere: Arbre_Binaire_Character.Piles_Cle.T_Pile;
            begin
                if not Est_Nul_AG(Noeud_Cle) then --Si la clé Cle existe
                    Arbre_Binaire_Character.Piles_Cle.Affecter_Pile(Desc_Mere,
Descendants_Noeud_Foret(Nodekey(Fils_Droit(Noeud_Cle)),F_Foret)); --
Desc_Mere reçoit l'ensemble des descendants de la "mère" qui existent dans tou
te la forêt
                    Arbre_Binaire_Character.Piles_Cle.Affecter_Pile(Desc_Pere,
Descendants_Noeud_Foret(Nodekey(Fils_Gauche(Noeud_Cle)),F_Foret));--
Desc_Pere reçoit l'ensemble des descendants du "père" qui existent dans toute
la forêt
                    Arbre_Binaire_Character.Piles_Cle.Supprimer_Element(Cle,De
sc_Pere); --Suppression de l'individu Cle de l'ensemble des enfants du Père
                    Arbre_Binaire_Character.Piles_Cle.Supprimer_Element(Cle,De
sc_Mere); --Suppression de l'individu Cle de l'ensemble des enfants de la Mère
                    --Concaténation des deux ensembles
                    while (not Arbre_Binaire_Character.Piles_Cle.Est_Vide(Desc
_Mere)) loop --Tant que l'ensemble Desc_Mere n'est pas vide
                        if not Arbre_Binaire_Character.Piles_Cle.Existe_Pile(A
rbre_Binaire_Character.Piles_Cle.Sommet(Desc_Mere),Desc_Pere) then --
Si l'élément n'existe pas déjà dans Desc_Pere alors
                            Arbre_Binaire_Character.Piles_Cle.Empiler(Desc_Per
e,Arbre_Binaire_Character.Piles_Cle.Sommet(Desc_Mere)); --
On empile le sommet de Desc_Mere dans Desc_Pere
                        else --
Si le sommet existe déjà dans Desc_Père et Desc_Mère c'est qu'il sagit soit d'
un frère soit d'une soeur. Il faut alors le supprimer de la pile.
                            Arbre_Binaire_Character.Piles_Cle.Supprimer_Elemen
t(Arbre_Binaire_Character.Piles_Cle.Sommet(Desc_Mere),Desc_Pere); --
Suppression de cet enfant commun de la pile Desc_Pere.
                        end if;
                        Arbre_Binaire_Character.Piles_Cle.Depiler(Desc_Mere);
                    end loop;
                    return Desc_Pere; --
Retourner la concaténation des deux ensembles, ou Desc_Pere si Desc_Mere est v
ide ou une pile vide si les deux sont vides.
                else --Si la clé Cle existe
```

```
                    Arbre_Binaire_Character.Piles_Cle.Initialiser(Desc_Pere);
--Pile vide
                    return Desc_Pere; --Retrouner une pile vide
                end if;
            end Half_Siblings;
        Ens:Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if Foret.Est_Vide(F_Foret) or Est_Nul_AG(AG) then --
Si la forêt ou l'arbre sont vides
                Put_Line("Impossible de chercher! L'arbre ou la forêt sont vid
es."); New_Line;
            else
                Arbre_Binaire_Character.Piles_Cle.Initialiser(Ens); --
Initialisation de l'ensemble de clés
                Arbre_Binaire_Character.Piles_Cle.Affecter_Pile(Ens,Half_Sibli
ngs(Cle,AG,F_Foret)); --
Ens reçoit l'ensemble des clés ayant au moins un parent en commun
                if not Arbre_Binaire_Character.Piles_Cle.Est_Vide(Ens) then --
Si la clé a bien des demi-soeurs ou demi-frères alors
                    Put("Les demi-frères et demi-
soeurs de la clé" & Integer'Image(Cle) & " sont : ");
                    Afficher_Ensemble2(Ens); --Affichage des clés
                    New_Line;
                    New_Line;
                else
                    Put_Line("Cette clé n'a ni des demi-soeurs ni des demi-
frères");New_Line;
                end if;
            end if;
        end Half_Sibling_Foret;

        procedure Multiplier_10_Foret(F_Foret: in out Foret.T_Pile) is
            Foret_V:Foret.T_Pile;
            AGForet: T_Branch;
            begin
                if Est_Vide(F_Foret) then
                    Put_Line("La forêt est vide!"); New_Line;
                else
                    Affecter_Pile(Foret_V,F_Foret);
                    while not Est_Vide(Foret_V) loop
                        AGForet:=Sommet(Foret_V);
                        Multiplier_10(AGForet);
                        Affecter_Pile(Foret_V,Next_Pile(Foret_V));
                    end loop;
                end if;
            end Multiplier_10_Foret;
```

```
                                        --AJOUT--

    function New_Key_Interval(Cle: in Integer; Predecesseur: in Integer; AG: i
n T_Branch) return Arbre_Binaire_Character.Piles_Cle.T_Pile is
        begin
            return NewKeyInterval(Cle,Predecesseur,AG);
        end New_Key_Interval;

    procedure Ajouter_Ancetre(Ancetre: in Integer; Sexe: in Character; Descend
ant: in Integer; AG: in out T_Branch) is
        begin
            Ajouter2(Ancetre,Sexe,Descendant,AG);
        end Ajouter_Ancetre;



                                    --MODIFICATION--

    procedure Modifier_Cle_Racine_AG(Cle: in Integer; AG: in out T_Branch) is
        begin
            Modifier_Cle_Racine(Cle,AG);
        end Modifier_Cle_Racine_AG;

    procedure Modifier_Cle_AG(Ancetre,NewAncetre: in Integer; AG: in out T_Bra
nch) is
        begin
            Modifier_Cle(Ancetre,NewAncetre,AG);
        end Modifier_Cle_AG;

    procedure Modifier_Sexe_AG(Ancetre: in Integer;NewSexe:in Character; AG: i
n out T_Branch) is
        begin
            Modifier_Donnee(Ancetre,NewSexe,AG);
        end Modifier_Sexe_AG;



                                    --SUPPRESSION--

    procedure Supprimer_Famille(Descendant: in Integer; AG: in out T_Branch) i
s
        begin
            Supprimer_Cle_ET_Fils(Descendant,AG);
        end Supprimer_Famille;



                                    --AFFICHAGE--

    procedure Afficher_AG_A_Partir(Descendant: in Integer; AG: in T_Branch) is
        begin
```

```
            Afficher_A_Partir(Descendant,AG);
        end Afficher_AG_A_Partir;

    procedure Afficher_AG(AG: in T_Branch) is
        Depth_AG: constant Integer :=Depth(AG);
        begin
            Put(" 0       ");
            for i in 1..(Depth_AG-1) loop
                Put(Integer'Image(i));
                Put("  ");
            end loop;
            Put("   Générations");New_Line;
            Put("-");
            for i in 1..Depth_AG loop
                Put("----");
            end loop;
            Put("--------------------");
            New_Line;
            Afficher(AG);
        end Afficher_AG;

end Arbre_Genealogique;
```