```
with Ada.Text_IO;                   use Ada.Text_IO;
with Ada.Integer_Text_IO;           use Ada.Integer_Text_IO;
with Ada.Unchecked_Deallocation;

package body Registre is

    procedure Free is
        new Ada.Unchecked_Deallocation (T_Registre, T_Access);

            --FONCTIONS/PROCEDURES SECONDAIRES--

    procedure CreateDate(D: in out T_Date; Jour: in Integer; Mois: in T_Mois;
Annee:in Integer) is
        begin
            D.Jour:=Jour;
            D.Mois:=Mois;
            D.Annee:=Annee;
        end CreateDate;

    procedure Afficher_Date (Date : in T_Date) is
        procedure Afficher_Deux_Positions (Nombre : in Integer) is
            begin
                Put (Nombre / 10, 1);
                Put (Nombre mod 10, 1);
            end Afficher_Deux_Positions; --
Pour afficher un jour ou un mois dans le format d'une date
        begin
            Afficher_Deux_Positions (Date.Jour);
            Put ('/');
            Afficher_Deux_Positions (T_Mois'pos (Date.Mois) + 1);
            Put ('/');
            Afficher_Deux_Positions (Date.Annee / 100);
            Afficher_Deux_Positions (abs(Date.Annee) mod 100);
            if Date.Annee<0 then
                Put(" BC");
            end if;
        end Afficher_Date;


                        --TESTS--
    procedure Init_RG(RG: in out T_Access) is
        begin
            RG:= new T_Registre;
        end Init_RG;
    procedure Start_RG(Cle: in Integer; Reg: in out T_Access) is
        begin
            Reg:= new T_Registre;
            Reg.all.Cle:=Cle;
```

```
    end Start_RG;

function Est_Vide_RG(Reg: in T_Access) return Boolean is
    begin
        return Reg=Null;
    end Est_Vide_RG;

function Existe_RG(Cle: in Integer;Reg: in T_Access) return Boolean is
    begin
        if Reg=Null then
            return False;
        else
            return (not Est_Vide_RG(Rech_Reg(Cle,Reg)));
        end if;
    end Existe_RG;


            --FONCTIONS/PROCEDURES ELEMENTAIRES--

function Name(Cle: in Integer; Reg:in T_Access) return Unbounded_String is
    RegKey: constant T_Access:=Rech_Reg(Cle,Reg);
    begin
        if RegKey/=Null then
            return RegKey.all.Nom_Complet;
        else
            return To_Unbounded_String("");
        end if;
    end Name;

function BirthD(Cle: in Integer; Reg: in T_Access) return T_Date is
    RegKey: constant T_Access:=Rech_Reg(Cle,Reg);
    D: T_Date;
    begin
        if RegKey/=Null then
            return RegKey.all.Date_Naissance;
        else
            CreateDate(D,01,JANVIER,0000);
            return D;
        end if;
    end BirthD;

function BirthY(Cle: in Integer; Reg: in T_Access) return Integer is
    RegKey: constant T_Access:=Rech_Reg(Cle,Reg);
    begin
        if Regkey/=Null then
            return RegKey.all.Date_Naissance.Annee;
        else
            return 0;
```

```
            end if;
        end BirthY;

    function BirthP(Cle: in Integer; Reg: in T_Access) return Unbounded_String
 is
        RegKey: constant T_Access:=Rech_Reg(Cle,Reg);
        begin
            if RegKey/=Null then
                return RegKey.all.Lieu_Naissance;
            else
                return To_Unbounded_String("");
            end if;
        end BirthP;

    procedure RG_Multiplier_10(RG: in out T_Access) is
        begin
            if Est_Vide_RG(RG) then
                Null;
            else
                RG.all.CLe:=RG.all.Cle*10;
                RG_Multiplier_10(RG.all.Suivant); --
Multiplier toutes les clés par 10
            end if;
        end RG_Multiplier_10;



                --FONCTIONS/PROCEDURES DE RECHERCHE--

    function Rech_Reg(Cle: in Integer;Reg: in T_Access) return T_Access is
        Parcours:T_Access;
        begin
            Parcours:=Reg;
            while Parcours/=Null and then Parcours.all.Cle/=Cle loop --
Tant qu'on n'est toujours pas arrivé ni à la clé recherchée ni à la fin du reg
istre
                Parcours:=Parcours.all.Suivant; --
Incrémentation du pointeur dans le registre complet
            end loop;
            return Parcours; --Récupérer le registre de la clé
        end Rech_Reg;



                    --AJOUT/SUPPRESSION--

    procedure AddKey(Cle: in Integer; Reg: in out T_Access) is
        Parcours1,Parcours2:T_Access;
        begin
            if Reg=Null then
```

```
                Reg:=new T_Registre;
                Reg.all.Cle:=Cle; --Nouveau registre initialisé par la Clé Cle
            elsif Rech_Reg(Cle,Reg)/=Null then
                Null;
            else
                Parcours1:=Reg;
                while Parcours1.all.Suivant/=Null and then Parcours1.all.Suiva
nt.all.Cle<Cle loop --
Pour insérer la clé dans le bon endroit afin que les clés soient triées dans l
'ordre croissant et assurer une recherche efficace des clés.
                    Parcours1:=Parcours1.all.Suivant;
                end loop;
                Parcours2:=Parcours1.all.Suivant; --
Garder le reste du registre
                Parcours1.all.Suivant:=new T_Registre;
                Parcours1.all.Suivant.all.Cle:=Cle; --
Insérer la clé à sa place
                Parcours1.all.Suivant.all.Suivant:=Parcours2; --
Rattacher les deux parties du registre
            end if;
        end AddKey;

    procedure Delete_RG(Cle:in Integer; Reg: in out T_Access) is
        Parcours,A_Supprimer:T_Access;
        begin
            if Reg=Null then
                Null;
            elsif Rech_Reg(Cle,Reg)=Null then
                Null;
            else
                if Reg.all.Cle=Cle then
                    Reg:=Reg.all.Suivant;
                else
                    Parcours:=Reg;
                    while Parcours.all.Suivant/= Null and then Parcours.all.Su
ivant.all.Cle/=Cle loop
                        Parcours:=Parcours.all.Suivant;
                    end loop;
                    A_Supprimer:=Parcours.all.Suivant;
                    Parcours.all.Suivant:=Parcours.all.Suivant.all.Suivant;
                    Free(A_Supprimer);
                end if;
            end if;
        end Delete_RG;

    procedure Detruire_RG(RG: in out T_Access) is
        begin
            if RG /= Null then
```

```
                Detruire_RG(RG.all.Suivant);
                Free (RG); --Detruction du registre
            else
                Null;
            end if;
        end Detruire_RG;


                --MODIFICATIONS--

    procedure ModifyKey(Cle: in Integer; NewCle: in Integer; Reg: in out T_Acc
ess) is
        KeyReg:T_Access;
        begin
            if Reg=Null then
                Reg:=new T_Registre;
                Reg.all.Cle:=NewCle;
            else
                KeyReg:=Rech_Reg(Cle,Reg);
                if KeyReg/=Null then
                    KeyReg.all.Cle:=NewCle;
                else
                    Null;
                end if;
            end if;
        end ModifyKey;

    procedure AddName(Cle: in Integer; Nom: in Unbounded_String; Reg:in out T_
Access) is
        KeyReg:T_Access;
        begin
            if Reg=Null then
                Reg:= new T_Registre;
                Reg.all.Cle:=Cle;
                Reg.all.Nom_Complet:=Nom;
            else
                KeyReg:=Rech_Reg(Cle,Reg);
                KeyReg.all.Nom_Complet:=Nom;
            end if;
        end AddName;

    procedure AddBirthD(Cle: in Integer; Jour: in Integer; Mois: in T_Mois; An
nee:in Integer; Reg: in out T_Access) is
        KeyReg:T_Access;
        begin
            if Reg=Null then
                Reg:= new T_Registre;
                Reg.all.Date_Naissance.Jour:=Jour;
```

```
                Reg.all.Date_Naissance.Mois:=Mois;
                Reg.all.Date_Naissance.Annee:=Annee;
                Reg.all.Age:=2019 - Annee;
            else
                KeyReg:=Rech_Reg(Cle,Reg);
                KeyReg.all.Date_Naissance.Jour:=Jour;
                KeyReg.all.Date_Naissance.Mois:=Mois;
                KeyReg.all.Date_Naissance.Annee:=Annee;
                KeyReg.all.Age:=2019 - Annee;
            end if;
        end AddBirthD;

    procedure AddBirthP(Cle: in Integer; Lieu: in Unbounded_String; Reg: in ou
t T_Access) is
        KeyReg:T_Access;
        begin
            if Reg=Null then
                Reg:= new T_Registre;
                Reg.all.Lieu_Naissance:=Lieu;
            else
                KeyReg:=Rech_Reg(Cle,Reg);
                KeyReg.all.Lieu_Naissance:=Lieu;
            end if;
        end AddBirthP;

    procedure Ajouter_Conjoint(Cle,Conjoint: in Integer; RG: in out T_Access)
is
        KeyReg:T_Access;
        Ens:arbre_genealogique.Arbre_Binaire_Character.Piles_Cle.T_Pile;
        begin
            if RG=Null then
                Put_Line("Vide!");
            else
                KeyReg:=Rech_Reg(Cle,RG);
                arbre_genealogique.Arbre_Binaire_Character.Piles_Cle.Affecter_
Pile(Ens,KeyReg.all.Conjoints);
                arbre_genealogique.Arbre_Binaire_Character.Piles_Cle.Empiler(E
ns,Conjoint);
            end if;
        end Ajouter_Conjoint;

end Registre;
```