```
with Ada.Text_Io, Ada.Integer_Text_Io;
use  Ada.Text_Io, Ada.Integer_Text_Io;

procedure Sudoku is
   --------------- begin définitions d'une grille ----------------
   subtype Symbole is Integer range 0..9;
   type Grille is array(1..9, 1..9) of Symbole;
   type Tbl9   is array(Symbole'Range) of Symbole;
   type TblSol is array(Symbole'Range) of Boolean;

   procedure Get(G : in out Grille) is
   begin
      for X in Grille'Range(1) loop
         for Y in Grille'Range(2) loop
            Get(G(X, Y));
         end loop;
      end loop;
   end;
   --------------- end   définitions d'une grille ----------------

   --------------- begin définitions des Piles -------------------
   type Pile is record
      Head : Integer;
      Data : Tbl9;
   end record;

   PileVide : constant Pile := (0, (others => 0));

   function Sommet(P : in Pile) return Symbole is
   begin
      return P.Data(P.Head);
   end;

   procedure Empiler(P : in out Pile; V : in Symbole) is
   begin
      P.Head := P.Head + 1;
      P.Data(P.Head) := V;
   end;

   procedure Depiler(P : in out Pile) is
   begin
      P.Head := P.Head - 1;
   end;

   function Vide(P : in Pile) return Boolean is
   begin
      return P.Head = 0;
   end;

   procedure Put(P : in Pile) is
      Pp : Pile := P;
   begin
      if not Vide(Pp) then
         Depiler(Pp);
         Put(Pp);
         Put(Sommet(P), 2);
      end if;
   end;
   --------------- end   définitions des Piles -------------------


   --------------- begin vos réponses ici...   -------------------
```



```
   ------------------------------------------------------
   ------------------------------------------------------
   -- Question 1
   procedure Put(G : in Grille) is
   begin
      for X in Grille'Range(1) loop
         for Y in Grille'Range(2) loop
            Put(G(X, Y), 2);
         end loop;
         New_Line;
      end loop;
   end;

   -- Question 2
   function Intersection(P1, P2 : in Pile) return Pile is
      P, Ptmp : Pile := PileVide;
      P11 : Pile := P1;
      P22 : Pile := P2;
   begin
      while not Vide(P11) and not Vide(P22) loop
         if Sommet(P11) = Sommet(P22) then
            Empiler(Ptmp, Sommet(P11));
            Depiler(P11);
            Depiler(P22);
         elsif Sommet(P11) < Sommet(P22) then
            Depiler(P22);
         else
            Depiler(P11);
         end if;
      end loop;

      while not Vide(Ptmp) loop
         Empiler(P, Sommet(Ptmp));
         Depiler(Ptmp);
      end loop;

      return P;
   end;

   -- Question 3
   procedure RemplirPile(Sol : in TblSol; P : out Pile) is
   begin
      P := PileVide;
      for I in Sol'Range loop
         if I /= 0 and Sol(I) then
            Empiler(P, I);
         end if;
      end loop;
   end;

   -- Question 4
   function TrouverSolCol(G : in Grille; X : in Integer) return Pile is
      Sol : TblSol := (others => True);
      P   : Pile;
   begin
      for Y in Grille'Range(2) loop
         Sol(G(X, Y)) := False;
      end loop;

      RemplirPile(Sol, P);
      return P;
   end;
```

```
   -- Question 5
   function TrouverSolLig(G : in Grille; Y : in Integer) return Pile is
      Sol : TblSol := (others => True);
      P   : Pile;
   begin
      for X in Grille'Range(1) loop
         Sol(G(X, Y)) := False;
      end loop;

      RemplirPile(Sol, P);
      return P;
   end;

   -- Question 6
   function TrouverSolReg(G : in Grille; X, Y : in Integer) return Pile is
      Sol : TblSol := (others => True);
      P   : Pile;
      X2, Y2 : Integer;
   begin
      X2 := (X - 1) / 3 * 3;
      Y2 := (Y - 1) / 3 * 3;
      for I in X2+1..X2+3 loop
         for J in Y2+1..Y2+3 loop
            Sol(G(I, J)) := False;
         end loop;
      end loop;

      RemplirPile(Sol, P);
      return P;
   end;

   -- Question 7
   function TrouverSol(G : in Grille; X, Y : in Integer) return Pile is
   begin
      return Intersection(TrouverSolReg(G, X, Y),
                          Intersection(TrouverSolLig(G, Y),
                                       TrouverSolCol(G, X)));
   end;

   -- Question 8
   procedure TrouverPremiereCaseVide(G : in Grille; X, Y : out Integer) is
   begin
      X := 0;
      Y := 0;
      for I in Grille'Range(1) loop
         for J in Grille'Range(2) loop
            if G(I, J) = 0 then
               X := I;
               Y := J;
               return;
            end if;
         end loop;
      end loop;
   end;

   -- Question 9
   procedure Solve(G : in Grille; Solved : in out Boolean; GSolved : in out Gril
le) is
      X, Y : Integer := 0;
      S    : Pile;
      Gtmp : Grille;
```



```
   begin
      TrouverPremiereCaseVide(G, X, Y);
      if X = 0 and Y = 0 then
         Solved  := True;
         GSolved := G;
         return;
      end if;
      S := TrouverSol(G, X, Y);
      Gtmp := G;
      while not Vide(S) loop
         Gtmp(X, Y) := Sommet(S);
         Solve(Gtmp, Solved, Gsolved);
         if Solved then
            return;
         end if;
         Depiler(S);
      end loop;
   end;

   -- Question 10
   -- 81 - nombre de valeurs déjà présentes.
   --------------- end   vos réponses ici...   --------------------

   G       : Grille;
   Gsolved : Grille;
   Solved  : Boolean := false;
--      P, P2 : Pile;
begin
--      P := PileVide;
--      Empiler(P, 1);   Empiler(P, 2);    Empiler(P, 5);    Empiler(P, 9);
--      P2 := PileVide;
--      Empiler(P2, 2);  Empiler(P2, 3); Empiler(P2, 5);    Empiler(P2, 8);
--      Put("P :  ");
--      Put(P); New_Line;
--      Put("P2 : ");
--      Put(P2); New_Line;
--      Put("P inter P2 : ");
--      Put(Intersection(P, P2)); New_Line;
--      Put("Intersection normale : 2 5"); New_Line;

   Get(G);
   Solve(G, Solved, Gsolved);
   if (Solved) then
      Put(Gsolved);
   else
      Put("*error*");
   end if;
end;
```