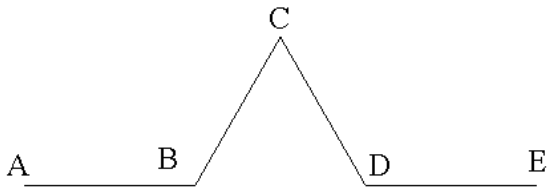
partie I : décomposition d'un segment [A,E] en 4 segments [A,B], [B,C], [C,D], [D,E]

Nous allons dans un premier temps découvrir une décomposition d'un segment de droite initial [A, E]. Le point A est de coordonnées (Ax, Ay) et E de coordonnées (Ex, Ey). Ce segment de droite est coupé en 3 parties égales, [A, B], [B, D] et [D, E]. Nous retirons le segment [B, D]. Nous ajoutons les segments [B, C] et [C, D] tels qu'ils soient les côtés d'un triangle équilatéral. La figure ci-dessous donne le résultat.



Les coordonnées d'un point P quelconque de la droite passant par les points A et E sont données par l'équation paramétrique : P = (1-t).A + t.E où t est un réel. C'est à dire : xp = (1-t) * Ax + t* Ex **et** yp = (1-t) * Ay + t* Ey.

- t=0 donne le point A
- t=1 donne le point E
- t=1/3 donne le point B
- t=2/3 donne le point D
- t=1/2 donne le point H (non dessiné), milieu du segment [A, E]

Nous pouvons donc calculer facilement les coordonnées des points B, D et H en connaissant uniquement les coordonnées de A et E. La projection orthogonale du point C sur le segment [A, E] donne le point H.
Soit N= (Ey-Ay, Ax-Ex) un vecteur orthogonal au segment [A, E]. Le point C est obtenu par C = H + (√3/6) N.

Question 1 (0,5pt) : Définir le type **point** qui contient deux coordonnées réelles.

Réponse : `type point = record x,y:real; end;`

Question 2 (0,5 pt) : Ecrire une procédure **calcule_point** qui calcule un point **P** à partir des points **A** et **E** et de **t**.

Réponse :
```
procedure calcule_point(A,E : point; t : real; var P : point);
begin
      P.x := (1-t)*A.x + t*E.x;
      P.y := (1-t)*A.y + t*E.y;
end;
```

Question 3 (0,5 pt) : Ecrire une procédure **calcule_N** qui calcule le vecteur **N** (vu comme un point) à partir de deux points **A** et **E**.

Réponse :
```
procedure calcule_N(A,E : point ; var N : point);
begin
      N.x := E.y - A.y;
      N.y := A.x - E.x;
end;
```

Question 4 (0,5 pt) : Ecrire une procédure **calcule_C** qui détermine le point **C**. Vous utiliserez **obligatoirement** les deux procédures précédentes.

Réponse:
```
procedure calcule_C(A , E :  point; var C : point);
var H, N : point;  K : real;
begin
      calcule_N(A, E, N);
      calcule_point(A, E, 0.5 , H);
      K := (1/6)*sqrt(3);
```

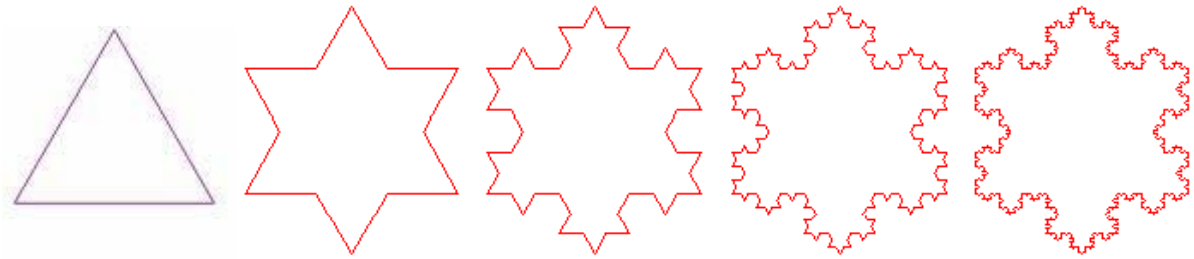
```
      C.x := H.x + K*N.x;
      C.y := H.y + K*N.y;
end;
```
(Pour moi 2 pts ici)

Partie II :

L'objectif est désormais de :

1) calculer les coordonnées géométriques d'un flocon de Koch qui est une fractale.
2) de les afficher à l'écran

La figure ci-dessous illustre notre explication. Le flocon de rang 0 est un triangle équilatéral (à gauche). Ce flocon initial est formé de trois points P1, P2, P3. Initialement nous avons P1 = (0,0), P2 = (1, 0) et P3 = (1/2, (√3/2). Nous appliquons la décomposition décrite en partie 1 à chacun des segments [P1, P2] [P2, P3] et [P3, P1]. Nous obtenons ainsi le flocon de rang 1 à droite du triangle. En ré-appliquant la décomposition aux segments obtenus, nous obtenons successivement les flocons de rang 2, 3 et 4 (de la gauche vers la droite du dessin).



Nous définissons le début du programme :

const MAX_POINT = 10000 ; (*MAX_POINT est le nombre maximum de points *)

```
type floc = record
               rang, nb_point : integer;
               t_point : array[1..MAX_POINT] of point;
            end;
```

Deux points ***consécutifs*** d'indice **i** et **i+1** du tableau **t_point** définissent un segment. Un champ **nb_point** permet de gérer explicitement le nombre de points du contour (3 initialement). Le **rang** du flocon est un entier initialisé à 0. Nous recopions dans la case **nb_point+1** du tableau le point de la case d'indice **1** pour « fermer » le contour. Ainsi tous les segments sont consécutivement codés dans le tableau. La procédure suivante initialise le flocon de rang 0 :

```
procedure init_flocon(var un_floc : floc);
begin
      un_floc.rang := 0;
      un_floc.nb_point := 3;
      with un_floc do
            begin
                  t_point[1].x := 0.0;
                  t_point[1].y := 0.0;
                  t_point[2].x := 1.0;
                  t_point[2].y := 0.0;
                  t_point[3].x := 0.5;
                  t_point[3].y := 0.5*sqrt(3);
                  t_point[4]   := t_point[1];
            end;
end;
```

Question 5 (1 pt) : Chaque segment décomposé génère **3 nouveaux points**. Si l'on décompose le segment entre les points indices **i** et **i+1** du tableau, il faut générer 3 places pour 3 nouveaux points **entre** ces deux indices. La fonction suivante "fait" de la place à partir d'un indice **i** et retourne le nouvel emplacement de l'ancien point de numéro i+1 qui est « décalé ».

```
function faire_place(var un_floc : floc; i : integer) : integer;

var k : integer;

begin
  with un_floc do
      begin
            for k := nb_point + 1 downto i+1 do
              begin
                  t_point[k+3]:= t_point[k];
            (* donner la valeur de k et dessiner le contenu tableau t_point ici *)
              end ;
            nb_point := nb_point+3;
      end;
  faire_place := i+4;
end;
```

Soit flocon de **rang** 0, avec **nb_point** = 3, et le tableau **t_point** initialisé et schématisé comme ci-dessous :

1	2	3	4	5	6	7	8	9	10	11	12	13
P1	P2	P3	P1									

Faire fonctionner « a mano » la fonction pour i = 1 en suivant l'indication en commentaire dans le code, donner la nouvelle valeur du champ **nb_point** et la valeur retournée par la fonction.

Réponse :

`k = 3 + 1 = 4`

1	2	3	4	5	6	7	8	9	10	11	12	13
P1	P2	P3	P1			P1						

`k = 3`

1	2	3	4	5	6	7	8	9	10	11	12	13
P1	P2	P3	P1		P3	P1						

`k = 2`

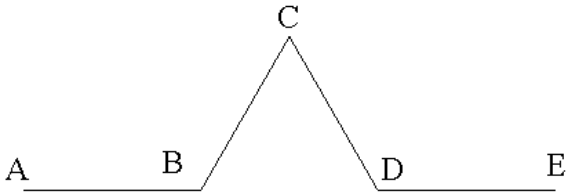
1	2	3	4	5	6	7	8	9	10	11	12	13
P1	P2	P3	P1	P2	P3	P1						

Le champ **nb_point** vaut 6 et la fonction retourne 5.

(Pour moi 3 pts ici)

Question 6 (1 pt) : Concevoir une fonction **ajoute_points** qui crée et met à jour les nouveaux points dans une variable de type floc pour un segment défini par l'indice i de son premier point. Cette fonction retournera l'indice du premier point du prochain segment à partir duquel de nouveaux points devront être ajoutés. On utilisera **obligatoirement** les variables locales A, B, C, D et E de type point pour que le code soit clair (voir première figure avec le segment décomposé) ainsi que la fonction **`faire_place`** ci-dessus.

Réponse avec rappel de la figure :




```
function ajoute_point(var un_floc : floc ; i : integer) : integer ;

var A, B, C, D, E : point ;
      new_depart : integer ;

begin
```

```
  with un_floc do
     begin
       A := t_point[i] ;
       E := t_point[i+1] ;
       new_depart := faire_place(un_floc, i );
       calcule_point(A, E, 1/3, B);
       calcule_point(A, E, 2/3, D);
       calcule_C(A, E, C);
       t_point[i+1] := B ;
       t_point[i+2] := C ;
       t_point[i+3] := D ;
     end;
  ajoute_point := new_depart;
end;
```

(Pour moi 4 pts ici)

Question 7 (1 pt) : Concevoir une procédure **`floc_r_2_r_plus_1`** qui calcule et met à jour les données quand un flocon passe d'un rang r au rang r+1. Vous utiliserez obligatoirement une boucle repeat until.

Réponse : la difficulté est de savoir déterminer simplement le critère d'arrêt. Il faut se rappeler que le premier point est aussi le dernier. Quand la fonction ajoute_point retourne l'indice du prochain premier indice de point pour le segment suivant, il suffit de tester si ces coordonnées sont les mêmes que celles du premier point du tableau.

Réponse :

```
procedure floc_r_2_r_plus_1(var un_floc : floc) ;
var stop, i, next : integer ;
begin
   stop := 0;
   i := 1;
   repeat
      next := ajoute_point(un_floc, i);
      with un_floc do
         begin
            if ((t_point[i].x = t_point[next].x) and
                (t_point[i].y = t_point[next].y))
            then
                begin
                    stop := 1 ;
                    rang := rang + 1;
                end ;
         end;
   until (stop = 1);
end;
```

(Pour moi 5 pts ici)

Question 8 (1 pt) : On dispose de la procédure définie par **procedure dessine_segment(A,B : point) ;** qui trace à l'écran le segment entre les points A et B. Concevoir le programme principal avec les déclarations de variables globales qui calcule les coordonnées d'un flocon de rang 4 et qui le dessine ensuite.

Réponse :

```
program floc_4 ;
var rang, i : integer ;
    un_floc : floc;
begin
   init_flocon(un_floc);
   for rang := 1 to 4 do
      floc_r_2_r_plus_1(un_floc);
```

```
   with un_floc do
      for i := 1 to nb_point do
         dessine_segment(t_point[i], t_point[i+1]);
end.
```

(Pour moi 6 pts ici)