Pour le Final :

L'objectif est in fine de calculer la surface de polygones fermés. Un polygone est codé sous la forme d'une suite de points formant les segments consécutifs des polygones.

Partie I :

Nous commencerons par mettre au point une fonction qui calcule la surface d'un trapèze. Soit deux points A et B de coordonnées respectives (x1,y1) et (x2,y2), nous définissons la surface du trapèze sous le segment par la formule suivante : Surface=(x2-x1)*(y1+y2)/2. Cette formule ne devra pas être modifiée.

Q1 (0,5 pt) : écrire la fonction dont l'en-tête est :

```
function surf_trapez(x1,y1,x2,y2 :real) :real ;
```

Qui calcule la surface selon la formule ci-dessus.

Réponse :

```
function surf_trapez(x1,y1,x2,y2 :real) :real ;

begin
     surf_trapez :=(x2-x1)*(y1+y2)/2 ;
end
```

Q2 (0.5 pt) : Calculer la surface S1 du trapèze donné par les points A=(x1=1, y1=1), et B=(x2=2,y2=2). Soit S1 :=`surf_trapez(1,1,2,2);`

Réponse : en appliquant la fonction on trouve : S1=(2-1)*(2+1)/2=+3/2

Q3 (0.5 pt) : Calculer la surface S2 du trapèze donné par les points B=( x1=2,y1=2) et C=(x2=1,y2=3). Soit S2 :=`surf_trapez(1,2,2,3);` . Cette surface est négative, c'est normal bien que cela semble absurde de prime abord.

Réponse : en appliquant la fonction on trouve : S2=(1-2)*(3+2)/2=-5/2

Q4 (0.5 pt) : Calculer S=S1+S2. Au signe près est-ce que S correspond à la surface du triangle (A,B,C) ? Dessinez les points, et hachurez les surfaces S1 et S2.

Réponse : On trouve S=-1, au signe près c'est bien la surface du triangle (A,B,C) qui a un grand côté de taille 2.

Partie II :

Un polygone est définit sous la forme d'un tableau de n lignes et de deux colonnes. Voici le début du programme pascal :

```
Program calcule_surface_polygone;

Const MAX_NB_POINTS=32;
Type polygone= array[1..MAX_NB_POINTS,1..2] of real ;
var
    poly : polygone;
    nb_points_utiles : integer ;
```

Une ligne i correspond aux coordonnées du point numéro i. Pour une ligne i, poly[i,1] donne la coordonnée sur l'axe des x et poly[i,2] donne la coordonnée sur l'axe des y.

Deux points consécutifs d'indice i et i+1 dans le tableau définissent le segment entre les points i et i+1.

Le dernier segment qui ferme le polygone est **implicitement** définit entre le point numéro `nb_points` et le point numéro 1. On remarquera que les déclarations ci-dessus limitent le nombre de segments à 32.

La variable `nb_points_utiles` correspondra au nombre de points réellement utiles présents dans le tableau. On supposera que `nb_points_utiles` est inférieur ou égal à `MAX_NB_POINTS`.

Q5 (1,5 pts) : écrire la fonction qui calcule la surface d'un polygone en parcourant consécutivement ses segments. L'en-tête sera obligatoirement :

```
function surf_polygone(pol : polygone, nbpt : integer) :real ;
```

Pol contiendra le polygone et nbpt le nombre de points du polygone.

Réponse :

```
function surf_polygone(pol : polygone; nbpt) :real ;

var I : integer;
    x1, x2, y1, y1, surf : real;

begin
  surf:=0;
  for I:=1 to nbpt-1 do
      begin
        x1:= pol[i,1];
        y1:= pol[i,2];
        x2:= pol[I+1,1];
        y1:= pol[I+1,2];
        surf=surf+ surf_trapez(x1,y1,x2,y2) ;
      end
  x1:= pol[nbpt,1];
  y1:= pol[nbpt,2];
  x2:= pol[1,1];
  y1:= pol[1,2];
```

```
  surf=surf+ surf_trapez(x1,y1,x2,y2) ;
  surf_polygone:= surf ;
end
```

Q6 (0,5 pt) : Appliquer avec `nb_points_utiles=4` et le polygone suivant :

`poly=`

| | |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 1 | 3 |
| 0 | 2 |

**Au signe près** vous avez la bonne réponse si votre fonction est correcte.

Réponse : c'est la suite du triangle d'avant. C'est un carré de côté √2 mais à cause du sens de parcours le résultat est `surf=-2.`

Q7 (1,5 pt) : Vous avez pu remarquer que le sens de parcours des segments influe sur le signe du résultat, c'est-à-dire le signe de la surface. Si les segments sont définis dans le sens trigonométrique en parcourant le polygone alors la surface calculée est négative. S'ils sont définis dans le sens des aiguilles d'une montre alors la surface sera positive.
Ecrire une procédure reverse qui inverse le sens de parcours. L'en tête en sera obligatoirement :

```
Procedure rev_pol(var pol_reverse: polygone; pol: polygone; nbpt:integer);
```

Pol_reverse contiendra le polygone résultat, pol le polygone d'origine et nbpt le nombre de points du polygone d'origine.

Réponse :

```
Procedure rev_pol(var pol_reverse: polygone; pol: polygone; nbpt: integer);


Var I,indice :integer;

Begin
   For I:=1 to (nbpt div 2) do
      Begin
        Indice:= nbpt-i+1 ;
        pol_reverse[I,1]:= pol[Indice,1];
        pol_reverse[I,2]:= pol[Indice,2];
      end
end
```