Corrigé de l'épreuve facultative d'informatique de l'X - 2003 - MP/PC

Compte tenu de l'implémentation des tableaux en Maple, on introduit la fonction `longueur` suivante valable pour toute les questions :

```
longueur := proc(a :: array)
    return(nops(convert(a,list));
end;
```

Question 1 :

```
sudouest := proc(x1,y1,x2,y2)
    return( x1 <= x2 and y1 <= y2)
end;

nordouest := proc(x1,y1,x2,y2)
   return( x1 <= x2 and y1 >= y2)
end;

sudest := proc(x1,y1,x2,y2)
   return( x1 >= x2 and y1 <= y2)
end;

nordest := proc(x1,y1,x2,y2)
   return( x1 >= x2 and y1 >= y2)
end;
```

Question 2 : On prend soin de ne pas écraser les données en passant par une variable intermédiaire c. La fonction `echange` ne retourne rien, elle se contente de permuter des éléments dans les tableaux a et b.

```
echange := proc(a::array,b::array,i::nonnegint,j::nonnegint)
 local c;
   c:=a[i]; a[i]:=a[j]; a[j]:=c;
   c:=b[i]; b[i]:=b[j]; b[j]:=c;
   return()
end;
```


Question 3 : Dans l'exemple de l'énoncé, la partie sud-ouest de l'enveloppe est formée de 2 points qui sont P_1 et P_0.

Question 4 : On parcourt tous les points P_i à la recherche des points de la partie sud-ouest de l'enveloppe.
ligne 6 : on suppose que P_i appartient à cette partie,
ligne 7-11 : on parcourt tous les points P_j à la recherche d'un point qui serait au sud-ouest de P_i et qui serait différent de P_i ce qui signifierait dans ce cas que le point P_i n'appartient pas à la partie sud-ouest de l'enveloppe
ligne 12 : Si la variable `ok` est restée à `true`, c'est que P_i appartient à partie sud-ouest de l'enveloppe. On effectue la permutation demandée et on incrémente k.

```
frontiereSO := proc(a :: array, b :: array)
  local n, ok, i, j, k;
  n := longueur(a);
  k := 0;
  for i from 0 to n-1 do
     ok := true;
     for j from 0 to n-1 do
        if (sudouest(a[j],b[j],a[i],b[i])
            and (a[i]<>a[j] or b[i]<>b[j]))
             then ok := false; fi;
     od;
    if ok then echange(a,b,k,i); k := k+1 fi;
  od;
  return(k);
end;
```

Question 5 : Les points définissant la partie nord-ouest de l'enveloppe dans l'exemple de l'énoncé sont les points P_0 et P_{11}. Pour écrire la fonction `frontiereNO`, il suffit de remplacer la fonction `sudouest` du programme précédent par la fonction `nordouest`.

```
frontiereNO := proc(a :: array, b :: array)
  local n, ok, i, j, k;
  n := longueur(a);
  k := 0;
  for i from 0 to n-1 do
     ok := true;
     for j from 0 to n-1 do
        if (nordouest(a[j],b[j],a[i],b[i])
            and (a[i]<>a[j] or b[i]<>b[j]))
             then ok := false; fi;
     od;
     if ok then echange(a,b,k,i); k := k+1 fi;
  od;
  return(k);
end;
```

Question 6 : Les fonctions `frontiereSE` et `frontiereNE` s'écrivent tout aussi simplement sous la forme suivante :

```
frontiereSE := proc(a :: array, b :: array)
  local n, ok, i, j, k;
  n := longueur(a);
  k := 0;
  for i from 0 to n-1 do
     ok := true;
     for j from 0 to n-1 do
        if (sudest(a[j],b[j],a[i],b[i])
            and(a[i]<>a[j] or b[i]<>b[j]))
               then ok := false; fi;
     od;
     if ok then echange(a,b,k,i); k := k+1 fi;
  od;
  return(k);
end;
```

```
frontiereNE := proc(a :: array, b :: array)
  local n, ok, i, j, k;
  n := longueur(a);
  k := 0;
  for i from 0 to n-1 do
     ok := true;
     for j from 0 to n-1 do
        if (nordest(a[j],b[j],a[i],b[i])
            and (a[i] <> a[j] or b[i] <> b[j]))
               then ok := false; fi;
     od;
     if ok then echange(a,b,k,i); k := k+1 fi;
  od;
  return(k);
end;
```

Question 7 : Il ne faut pas perdre de vue que les déplacements se font parallèlement aux axes des abscisses et des ordonnées. On va parcourir l'enveloppe dans le sens trigonométrique positif en partant de la partie sud-ouest.
On écrit d'abord deux fonctions de déplacement pour passer d'un point de coordonnée (x_1, y_1) à un point de coordonnées (x_2, y_2), l'une ,`deplaceSuivantX`, qui effectue d'abord un déplacement parallèlement à l'axe des abscisses et l'autre, `deplaceSuivantY`, qui effectue d'abord un déplacement parallèlement à l'axe des ordonnées.

```
deplaceSuivantX = proc(x1,y1,x2,y2)
    moveTo(x1,y1);
    lineTo(x2,y1);
    lineTo(x2,y2);
 end;

deplaceSuivantY = proc(x1,y1,x2,y2)
    moveTo(x1,y1);
    lineTo(x1,y2);
    lineTo(x2,y2);
 end;
```

```
13 enveloppe := proc()
14  local i;
15  for i from 0 to nSO-2 do
16      deplaceSuivantX(aSO[i],bSO[i],aSO[i+1],bSO[i+1]);
17  od;
18  deplaceSuivantX(aSO[nSO-1],bSO[nSO-1],aSE[0],bSE[0]);
19  for i from 0 to nSE-2 do
20      deplaceSuivantY(aSE[i],bSE[i],aSE[i+1],bSE[i+1]);
21  od;
22  deplaceSuivantY(aSE[nSE-1],bSE[nSE-1],aNE[nNE-1],bNE[nNE-1]);
23  for i from nNE-1 to 1 by -1 do
24      deplaceSuivantX(aNE[i],bNE[i],aNE[i-1],bNE[i-1]);
25  od;
26  deplaceSuivantX(aNE[0],bNE[0],aNO[nNO-1],bNO[nNO-1]);
27  for i from nNO-1 to 1 by -1 do
28      deplaceSuivantY(aNO[i],bNO[i],aNO[i-1],bNO[i-1]);
29  od;
30  deplaceSuivantY(aNO[0],bNO[0],aSO[0],bSO[0]);
31  end;
```

Question 8 : La réponse doit-être oui mais je ne vois pas de cas où l'enveloppe peut produire un polygone croisé.