Énoncé

On définit une procédure `boustrophedon` par le code Maple suivant

```
boustrophedon := proc(n)
  local B,i,j;
  B:= array(0..n,0..n);
  for i from 0 to n do
    for j from 0 to n do
      B[i,j] := 0;
    od;
  od;

  B[0,0] :=1;

  for i from 1 to n do
    if i mod 2 = 1 then
      j:=1;
      while j<= i do
        B[i,j] := B[i-1,j-1] + B[i,j-1];
        j:= j+1;
      od;
    else
      j:=i-1;
      while j>= 0 do
        B[i,j] := B[i-1,j] + B[i,j+1];
        j:= j-1;
      od;
    fi;
  od;
  return(eval(B));
end proc:
```

1. On suppose que l'on a exécuté `A:=boustrophedon(5)`. Former le tableau des valeurs `A[i,j]` où `i` représente une ligne et `j` une colonne.

2. On définit une famille de nombre $b(i,j)$ pour $i \in \mathbb{N}$ et $j \in \{0, \cdots, i\}$ par :

$$b(0,0) = 1$$

$$\forall i \in \mathbb{N} : \begin{cases} i \text{ impair } \Rightarrow b(i,0) = 0 \\ i \text{ pair } \Rightarrow b(i,i) = 0 \end{cases}$$

$$\forall i \in \mathbb{N} : \left.\begin{array}{r} i \text{ impair} \\ 0 < j \leq i \end{array}\right\} \Rightarrow b(i,j) = b(i-1,j-1) + b(i,j-1)$$

$$\forall i \in \mathbb{N} : \left.\begin{array}{r} i \text{ pair} \\ 0 \leq j < i \end{array}\right\} \Rightarrow b(i,j) = b(i-1,j) + b(i,j+1)$$

Définir en code Maple une procédure récursive dont l'appel permet de renvoyer un $b(i,j)$. On ne demande pas ici de diagramme ou de langage usuel.

Corrigé

1. Le tableau est initialisé avec des valeurs nulles partout. Chaque ligne se construit à partir de la précédente, de gauche à droite pour les lignes impaires, de droite à gauche pour les lignes paires. Pour la ligne i, seules les valeurs de j entre 0 et i sont modifiées. Après l'appel de la procédure on obtient les valeurs suivantes

$i\backslash j$	0	1	2	3	4	5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	1	1	0	0	0	0
3	0	1	2	2	0	0
4	5	5	4	2	0	0
5	0	5	10	14	16	16

2. On reconnait dans la définition de la famille $b(i,j)$ les mêmes formules que dans la procédure `boustrophedon`. Il s'agit donc de former une procédure récursive produisant le même algorithme. Par exemple :

```
boustro_r := proc(i,j)
  if i=0 and j=0 then
    return 1;
  fi;
  if i mod 2 = 1 and j=0 then
     return 0;
  fi;
  if i mod 2 =0 and j=i then
     return 0
  fi;
  if i mod 2 = 1 then
    return boustro_r(i-1,j-1) + boustro_r(i,j-1);
  fi;
  if i mod 2 = 0 then
    return boustro_r(i-1,j) + boustro_r(i,j+1);
  fi;
end proc:
```