Algorithmes et Structures de Données
DS1 – Vendredi 15 novembre 2024
Durée 1h30 - Documents non autorisés

1. Sudoku (8 pts)

Le Sudoku est un jeu de stratégie composé d'une grille comportant n^2 lignes et n^2 colonnes et n^2 blocs (en général, n=3). Une grille est valide ssi :

- chaque colonne de la grille contient TOUS les nombres de 1 à n^2
- chaque ligne de la grille contient TOUS les nombres de 1 à n^2
- chaque bloc de n^2 cases contient également TOUS les nombres de 1 à n^2

Exemple pour n=3

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Pour vérifier la validité d'une grille, il faut tenir à jour un tableau de booléens de type `tab-bool` permettant de savoir si une valeur (comprise entre 1 et n^2) est présente dans la ligne, colonne ou bloc.

```
Const n = 3
Type tab-bool = tableau[1..n*n] de booléens
     Sudoku = tableau[1..n*n,1..n*n] d'entiers
```

3.1. Ecrire en pseudo-langage la fonction `vérifie-tab(t : tab-bool) : booléen`, qui vérifie que chaque booléen de `t` est égal à `vrai`. Elle renvoie `vrai` dans ce cas, `faux` sinon.

3.2. Ecrire en pseudo-langage la fonction `vérifie-ligne(s : sudoku ; l : entier) : booléen`, qui vérifie que la ligne `l` est valide.

3.3. Ecrire en pseudo-langage la fonction `vérifie-colonne(s : sudoku ; c : entier) : booléen`, qui vérifie que la colonne `c` est valide.

3.4. Ecrire en pseudo-langage la fonction `vérifie-bloc(s : sudoku ; i, j : entier) : booléen`, qui vérifie que le bloc (dont le coin supérieur gauche est en `i,j)` est valide.

3.5. Ecrire en pseudo-langage la fonction `vérifie-sudoku(s : sudoku) : booléen`, qui vérifie que la grille complète du sudoku est valide.

3.6. On souhaite charger une grille de soduku à résoudre à partir du fichier texte de nom `nomfich`.
Chaque ligne du fichier est une ligne du sudoku dont les chiffres sont séparés par un `' '`.
Ecrire en pseudo-langage la procédure `charge-sudoku(E nomfich : chaine, S t : sudoku)`.

3.7. On souhaite sauvegarder la grille de sodoku résolue dans un fichier texte dont on demandera le nom à l'utilisateur. Chaque ligne du fichier est une ligne du sudoku dont les chiffres sont séparés par un `' '`.
Ecrire en pseudo-langage la procédure `sauvegarde-sudoku(E t : sudoku)`.

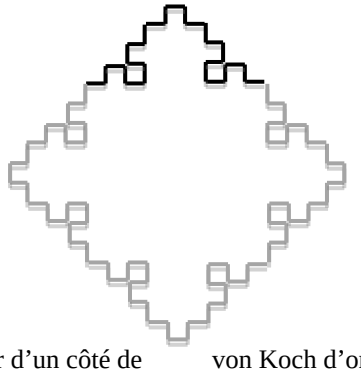
2. Fonctions récursives et pile (6 pts)

```
Fonction z(n : entier) : entier
Var res : entier
Début
Si n>0
      alors res←n-1
      sinon res←z(z(n+2){@1}){@2}
FinSi
Retourner(res)
Fin
```

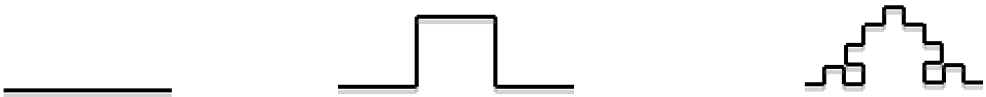
Simuler la pile sur l'appel écrire(z(-5)){@0} dans le programme principal.

3. Fractale : Carré de von Koch (6 pts)

On souhaite dessiner le carré de von Koch.
La figure ci-contre est obtenue avec quatre côtés de von Koch d'ordre n=2.



Pour dessiner un côté de carré de von Koch d'ordre n+1 à partir d'un côté de von Koch d'ordre n, il suffit de scinder en trois le côté et de remplacer le segment du milieu par trois segments comme le montre la figure ci-dessous.



n=0 n=1 n=2

Ecrire en pseudo-langage la **procédure récursive** `dessine-coté-VonKoch(E lg,n:entier)` qui permet de dessiner un côté du carré de longueur `lg` et de niveau `n`.
Comme outil de dessin, nous avons

- une procédure `trace-ligne(E lg:entier)` qui trace une ligne de longueur `lg` à partir du point courant (où se trouve le stylo) et dans la direction courante. `Trace-ligne` déplace le stylo.
- Une procédure `tourne(E d:entier)` qui tourne le stylo d'un angle donné par `d` en degré dans le sens trigonométrique si `d>0` (dans les sens inverse sinon).