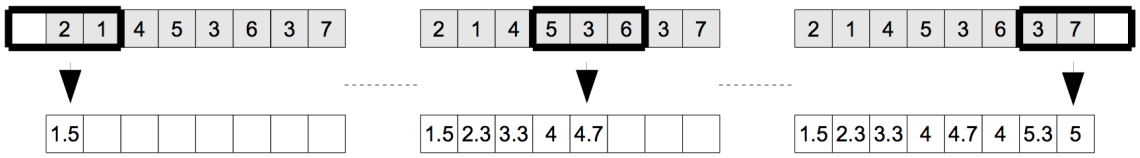
Algorithmes et Structures de Données
Mardi 4 Novembre 2025 - Correction

Exercice 1 : Lissage d'un tableau – 7 pts

On souhaite lisser une séquence de n valeurs en utilisant une fenêtre glissante de taille k ($1 < k < n$) pour moyenner les valeurs. Pour les premières et dernières valeurs, on ne prendra en compte que les valeurs réellement dans la fenêtre (cela réduit la taille de la fenêtre).
Exemple avec les 8 valeurs suivantes = 2 ; 1 ; 4 ; 5 ; 3 ; 6 ; 3 ; 7 et k=3



On utilisera le type t-val pour mémoriser les n valeurs et leur moyenne :

```
Const max = 20
Type  tab-val : tableau [1..max] de valeur
      valeur : Enregistrement
                 v, m : réel      {v est la valeur et m la valeur lissée}
               FinEnregistrement
```

1.1. Ecrire en pseudo-langage la procédure `lisser (E/S t : tab-val, E n, k : entier)` qui réalise ce lissage : les champs v des valeurs dans t sont initialisés et on veut calculer les champs m. On pourra écrire d'autres fonctions ou procédures pour simplifier l'écriture de cette procédure.

```
Procédure lisser (E/S t : tab-val, E n, k : entier)
Var i : entier
Début
Pour i←1 à n inc +1 faire
      t[i].m ← calculer-moyenne(t,n,i,k)
FinPour
Fin

Fonction calculer-moyenne(t : tab-val, n, i, k : entier) : réel
Var j, nb : entier
    s : réel
Début
s←0
nb←0
Pour j ←i-(k div 2) à i+(k - k div 2) - 1 inc +1 faire
      Si j>=1 et j<=n
            alors s←s+t[j].v
                  nb←nb+1
      FinSi
FinPour
Retourner(s/nb)
Fin
```

1.2. On souhaite sauvegarder le tableau t dans un fichier texte ayant le format suivant :

- n et k, séparés par un espace, sont écrits sur la première ligne
- les valeurs v de t, séparées par un espace, sont écrites sur la deuxième ligne (elles tiennent sur une ligne)
- et les moyennes m de t, séparées par un espace, sont écrites sur la troisième ligne (elles tiennent sur une ligne)

Pour l'exemple précédent :

```
8 3
2 1 4 5 3 6 3 7
1,5 2,3 3,3 4 4,7 4 5,3 5
```

Ecrire en pseudo-langage, **de façon optimisée**, une procédure qui sauvegarde le tableau `t` avec les valeurs `n` et `k` dans le fichier de nom `nomf`.

`Procédure sauvegarder (E nomf : chaine, t : tab-val, n, k : entier)`

On pourra utiliser la fonction `entier2chaine(i : entier) : chaine` qui transforme un entier `i` en une chaine de caractères (par exemple, l'entier `2,5` en la chaine "`2,5`").

```
Procédure sauvegarder (E nomf : chaine, t : tab-val, n, k : entier)
Var f : FT
    c, cv, cm : chaine
    i : entier
Début
f←Créer-fichier(nomf) {crée le ficher et l'ouvre en écriture}
c←concaténer(entier2chaine(n), ' ')
c←concaténer(c, entier2chaine(k))
ecrireChaine(f,c)
cv←''
cm←''
Pour i←1 à n inc +1 Faire
      cv←concaténer(cv, entier2chaine(t[i].v))
      cv←concaténer(cv, ' ')
      cm←concaténer(cm, entier2chaine(t[i].m))
      cm←concaténer(cm, ' ')
FinPour
ecrireChaine(f,cv)
ecrireChaine(f,cm)
Fermer(f)
Fin
```

Exercice 2 : Pile - 6 pts

Soient les 3 suites définies pour $n \geq 0$:

$u_0=1,\ u_{n+1}=2u_n+3v_n+w_n$

$v_0=2,\ v_{n+1}=u_n+v_n+2w_n$

$w_0=3,\ w_{n+1}=u_n+4v_n+w_n$

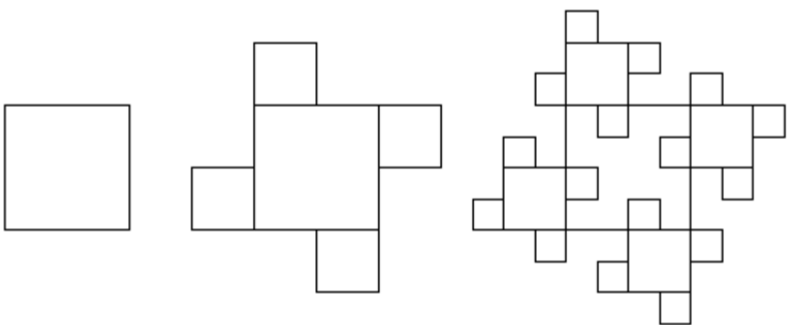
Simuler la pile sur l'appel `écrire(u(2)){@0}` dans le programme principal.

```
Fonction u (n : entier) : entier
Var res : entier
Début
  Si n=0 Alors res←1
         Sinon res←2*u(n-1){@1}+3*v(n-1){@2}+w(n-1){@3}
  FinSi
  Retourner(res)
Fin

Fonction v (m : entier) : entier
Var res : entier
Début
  Si m=0 Alors res←2
         Sinon res←u(m-1){@4}+v(m-1){@5}+2*w(m-1){@6}
  FinSi
  Retourner(res)
Fin

Fonction w (p : entier) : entier
Var res : entier
Début
  Si p=0 Alors res←3
         Sinon res←u(p-1){@7}+4*v(p-1){@8}+w(p-1){@9}
  FinSi
  Retourner(res)
Fin
```

~~@9, p=0 res=3~~
~~@8, m=0 res=2~~
~~@7, n=0 res=1~~
~~@3, p=1 res=1+4*2+3=12~~
~~@6, p=0 res=3~~
~~@5, m=0 res=2~~
~~@4, n=0 res=1~~
~~@2, m=1 res=1+2+2*3=9~~
~~@3, p=0 res=3~~
~~@2, m=0 res=2~~
~~@1, n=0 res=1~~
~~@1, n=1 res=2*1+3*2+3=11~~
~~@0, n=2 res=2*11+3*9+12=61~~

Exercice 2 : Dessin Récursif - 7 pts

On souhaite dessiner la figure dont on a mis ci-dessous uniquement les 3 premières étapes.



On dispose de la procédure `dessineCarre` (vue en TD) qui permet de dessiner un seul carré dont on passe en paramètre les coordonnées de ses 4 points (celui en bas à gauche en premier puis en tournant dans le sens des aiguilles d'une montre).
On souhaite écrire la procédure `Carre(E x, y, d : entier)` qui dessine la figure complète :

- `x` et `y` sont les coordonnées du sommet en bas à gauche du grand carré initial (en pixel),
- `d` est la longueur du côté du carré courant (en pixel),
- on fixera une constante `epsilon` à la taille du plus petit carré qu'on souhaite dessiner.

On rappelle que l'origine de l'écran est placée en haut à gauche (l'axe des abscisses vers la droite et l'axe des ordonnées vers le bas).

2.1. Expliquer le principe de la procédure `Carre` (principe récursif et critère d'arrêt).

2.2. Ecrire en pseudo-langage la procédure `Carre(E x, y, d : entier)`

```
Procédure Carre(E x, y, d : entier)
Const epsilon : 4
Début
Si d>epsilon
  Alors dessineCarre(x,y,x,y-d,x+d,y-d,x+d,y)
        Carre(x-(d div 2), y, d div 2)
        Carre(x, y-d, d div 2)
        Carre(x+d, ya-(d div 2), d div 2)
        Carre(x+(d div 2), y+(d div 2), d div 2)
FinSi
Fin
```