GIS3 — Structures de Données — Épreuve d'Avril 2018
Durée 2 heures — Aucun document autorisé

---

1 Petits exercices (6 points)

1.1 Tables de hachage

On considère l'insertion des clés $\square$, $\bigcirc$, $\triangle$, $*$ dans une table de hachage de $N = 13$ alvéoles. La table est gérée avec la technique du double hachage. Les valeurs de hachage des différents éléments sont les suivantes :

	h_1	h_2
$\square$	3	5
$\bigcirc$	7	7
$\triangle$	7	9
$*$	4	3

Question 1 [1 pt]. Insérer les clés dans la table. Donner un schéma de la table résultat. Indiquer les éventuelles collisions.

Question 2 [1 pt]. La fonction h_2 retourne un nombre impair dans tous les cas. Est-ce suffisant pour trouver un alvéole libre s'il en existe au moins un dans la table (justifier) ?

1.2 Arbres binaires de recherche

On rappelle la déclaration suivante :

```
struct abr {
    struct abr* gauche;
    int valeur;
    struct abr* droit;
};

#define NIL (struct abr*)0
```

Question 3 [2 pts]. Écrire une fonction qui retourne la somme des éléments de l'ABR qui lui est passé en paramètre.

1.3 Analyse de complexité

Question 4 [1 pt]. La fonction suivante calcule le produit de deux polynômes A et B de degré $n-1$. Le résultat est enregistré dans R. On s'intéresse à la fonction $f(n)$ qui mesure le nombre d'opérations arithmétiques effectuées sur les doubles. Que vaut $f(n)$ (justifier) ?

```
void mul_poly (double* R, double* A, double* B, int n)
{   int a, b;
    for (a = 0; a < 2*n-1; a++)
        R[a] = 0.0;
    for (a = 0; a < n; a++)
        for (b = 0; b < n; b++)
            R[a+b] = R[a+b] + A[a] * B[b];
}
```

Question 5 [1 pt]. Y a-t-il un meilleur des cas (justifier) ?

2 Problème (14 points)

On souhaite implanter une file de doubles au moyen d'une liste circulaire, suivant le principe illustré Figure 1. L'idée consiste à utiliser deux structures (`struct maillon` et `struct file`) adaptées de l'implantation des listes chaînées vue en cours et de mémoriser les deux extrémités de la file dans des pointeurs (`first_in` et `last_in`). On conseille aussi de mémoriser dans des champs distincts le nombre de maillons alloués et le nombre de maillons utilisés (c'est-à-dire le nombre d'éléments présents dans la file).

Question 6 [2 pts]. Donner la déclaration C des deux structures `struct maillon` et `struct file`.

Question 7 [2 pts]. Spécifier ces structures. Attention à bien préciser la signification des champs, à quoi on reconnaît la file vide ... Autorisez-vous les pointeurs à valoir NIL ? Suggestion : attendre la fin du problème avant de rédiger au propre la réponse à cette question.

Question 8 [2 pts]. Écrire un constructeur qui initialise une file à la file vide. Suggestion : allouer au minimum un maillon.

Question 9 [2 pts]. Écrire le destructeur.

Question 10 [2 pts]. Écrire l'action `enfiler`.

Question 11 [2 pts]. Écrire l'action (ou la fonction) `defiler`.

Question 12 [2 pts]. Écrire la fonction `file_vide`, qui retourne `true` si son paramètre est vide et `false` sinon.

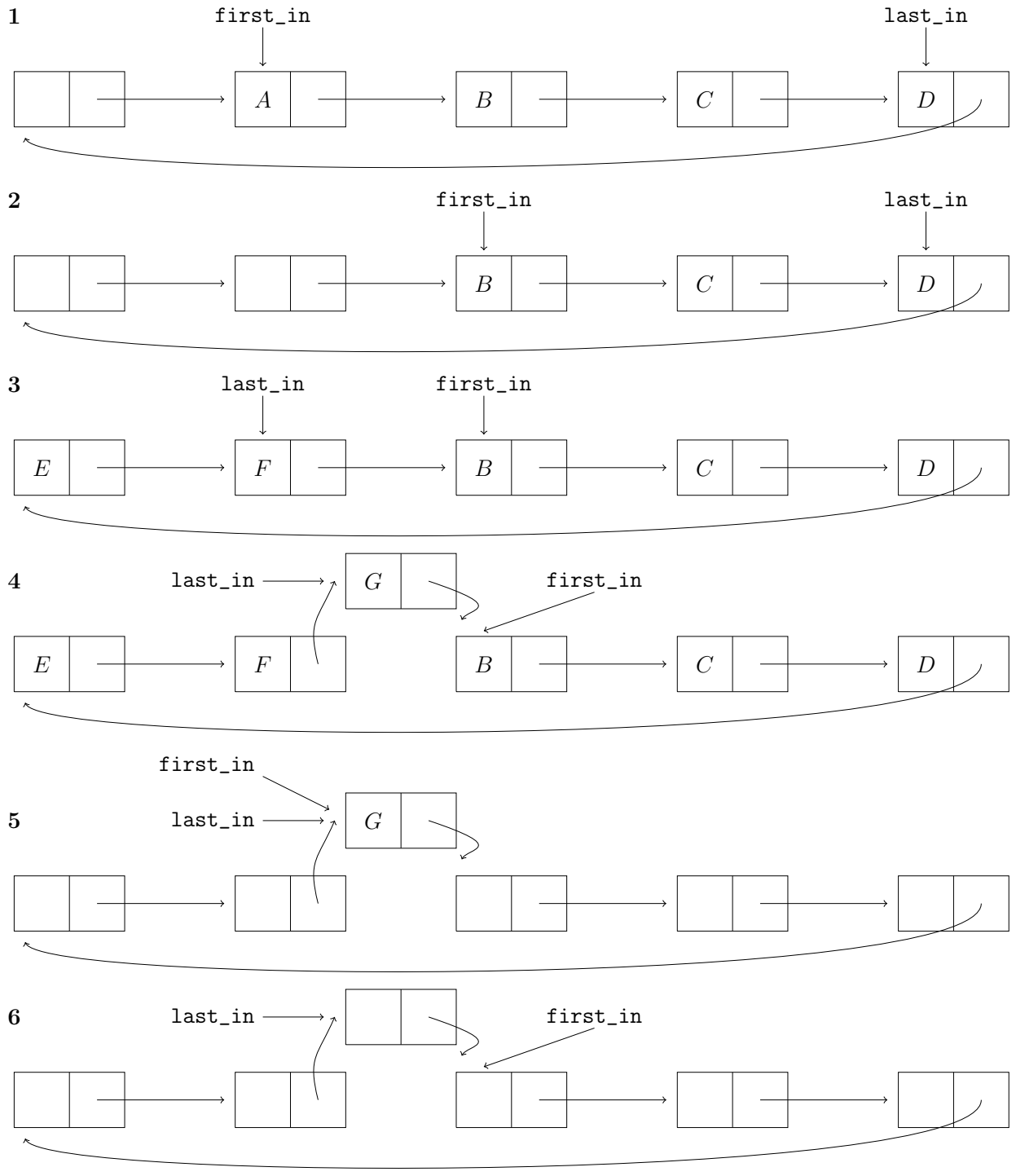

FIGURE 1 – **1**. File implantée sous la forme d'une liste circulaire de 5 maillons et comportant 4 éléments : A (premier entré), B, C et D (dernier entré). **2**. Après un défilage (= action de défiler). **3**. Après enfilage (= action d'enfiler) de E, puis de F. **4**. Pour enfiler G, un maillon supplémentaire doit être ajouté à la liste. **5**. Après 5 opérations de défilage, la file ne comporte plus qu'un élément. **6**. Encore un défilage et la file est vide