Nom : _____

Prénom : _____

Numéro étudiant : | | | | | | | | |

Allocation dynamique

13 mars 2019

A lire absolument :

1. L'objectif n'est pas d'apprendre la correction par cœur, mais de comprendre les mécanismes mis en œuvre. Cela vous permettra de vous adapter face à un problème nouveau.
2. En particulier, vous devez être capable de refaire l'intégralité du sujet, seul, sans aucune aide ni support.
3. Votre travail sera corrigé automatiquement par l'outil de correction automatique CAT. Cela implique que vous devez respecter scrupuleusement les consignes de chaque exercice. Faites très attention aux messages qu'il vous est demandé d'afficher. Un espace en trop, un saut de ligne en moins et l'exercice risque d'échouer.
4. L'enseignant voit votre activité sur le site, ainsi que l'historique de vos dépôts. Pensez à déposer votre travail régulièrement afin qu'il puisse vous apporter des conseils personnalisés.
5. Si l'enseignant vous demande de rendre votre travail sur papier, vous devez répondre directement sur le sujet en respectant absolument la zone prévue à cet effet. Tout ce qui se trouve en dehors de la zone sera ignoré.
6. Si le sujet contient un QCM, vous devez colorier les cases avec un stylo bleu ou noir. Les autres couleurs seront ignorées.
7. Chaque feuille est identifiée de manière unique. Vous pouvez donc rendre votre sujet avec les feuilles mélangées, mais il est préférable de les trier car cela vous permet de vérifier que vous n'en avez pas oublié une.
8. Si vous faites face à un problème, un bug, une erreur ou que vous souhaitez participer à l'amélioration de la plateforme, envoyez un mail à l'adresse suivante : support-cat@liste.parisnanterre.fr

Ne rien écrire dans cette zone

Allocation dynamique, première méthode

```
1 int * allocation1(int taille);
```

Écrire une fonction qui alloue dynamiquement un tableau d'entiers dont la taille est passée en argument. Cette fonction doit retourner l'adresse du premier élément du tableau. Pour cela, vous utiliserez la fonction malloc.

Allocation dynamique, deuxième méthode

```
1 int * allocation2(int taille);
```

Écrire une fonction qui alloue dynamiquement un tableau d'entiers dont la taille est passée en argument. Cette fonction doit retourner l'adresse du premier élément du tableau. Le tableau devra être initialisé avec des 0. Choisissez la bonne méthode.

Allocation dynamique, initialisation aléatoire

```
1 int * allocation3(int taille, int n);
```

Écrire une fonction qui alloue dynamiquement un tableau d'entiers dont la taille est passée en argument. Cette fonction doit retourner l'adresse du premier élément du tableau. Le tableau devra être initialisé avec des valeurs aléatoires comprises entre 0 et n inclus, un entier passé en argument. Choisissez la bonne méthode.

Ne rien écrire dans cette zone

Libération de mémoire

```
1 void liberation1(int * tab);
```

Écrire une fonction qui libère la mémoire d'un tableau alloué dynamiquement. L'adresse du premier octet de ce tableau est passée en paramètre.

Ne rien écrire dans cette zone

Concaténation de deux tableaux (version dynamique)

```
1 int * concatenation_2_tableaux(int * tab1, int taille1, int * tab2, int
    taille2);
```

Écrire une fonction qui attend en arguments deux tableaux d'entiers ainsi que leur taille (pas nécessairement la même) et retourne un nouveau tableau résultant de leur concaténation (premier tableau puis le second). Par exemple : 1|2|3|4 et 7|6|5|4|3|2 donne 1|2|3|4|7|6|5|4|3|2. Le tableau retourné doit être alloué dynamiquement.

Ne rien écrire dans cette zone

Sous tableau

```
1 int * sous_tableau(int * tab, int debut, int fin);
```

Écrire une fonction qui attend en arguments un tableau d'entiers ainsi qu'un indice de début et un indice de fin et retourne un nouveau tableau contenant les valeurs du premier tableau en partant de l'indice de début et jusqu'à l'indice de fin (inclus). Le tableau retourné doit être alloué dynamiquement sans gaspiller de mémoire. Par exemple : 1|2|3|4|7|6|5|4|3|2 avec $debut = 3$ et $fin = 5$ donne : 4|7|6. Si $fin < debut$, retourner $NULL$.

Ne rien écrire dans cette zone

Allocation à deux dimensions

```
1 int ** allocation_2_dimensions(int largeur, int hauteur);
```

Écrire une fonction qui alloue dynamiquement une matrice (pas nécessairement carrée) dont la largeur et la hauteur sont passées en arguments. La case de coordonnées (i, j) doit contenir la valeur $i * j$. Attention, lors de l'utilisation d'une matrice, on considère que la première coordonnée est l'abscisse, la seconde l'ordonnée. Ainsi la case de coordonnée $(2, 4)$ d'une matrice *mat* sera adressée par *tab*[2][4] et non *tab*[4][2].

Ne rien écrire dans cette zone

Liberéééééééééé, delivréééééééé

```
1 void liberation_matrice(int ** mat, int taille);
```

Écrire une fonction permettant de libérer la mémoire d'un tableau à deux dimensions alloué dynamiquement. La taille de la première dimension étant passée en argument.

Ne rien écrire dans cette zone