

Université Côte d'Azur Structure de Données PO2 SITE

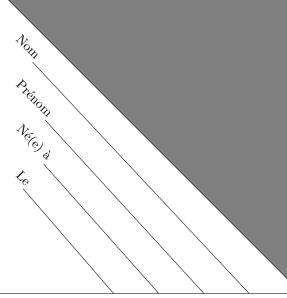
Épreuve de contrôle terminal du Mardi 15 décembre 2020

Durée: 1 heure

Tous documents autorisés. Il est interdit d'accéder à internet.

Note

Toutes les questions sont indépendantes. Tous les algorithmes devront être écrits en pseudo code. La notation est donnée à titre indicatif.



Il est de votre responsabilité de rabattre le triangle grisé et de le cacheter au moyen de colle, agrafes ou papier adhésif. Si ne vous le faites pas, vous acceptez implicitement que votre copie ne soit pas anonyme.

Exercice 1 Execution d'algorithme (4 points)

Sujet A

1:

2: 3:

4:

Soit l'algorithme suivant :

```
\begin{array}{l} algo\left(T[\,]\,,\ n\right)\ \{\\ pour\ (i\ de\ 2\ \grave{a}\ n)\ \{\\ a<-\ T[\,i\,]\\ j<-\ i\\ tant\ que\ (j>1\ et\ T[\,j\,-1]< a)\ \{\\ T[\,j\,]<-\ T[\,j\,-1]\\ j<-\ j\,-1\\ \}\\ T[\,j\,]<-\ a\\ \}\\ \end{array}
```

1. Exécutez cet algorithme et donnez le nombre de modifications effectuées pour le tableau suivant :

1	2	3	4	5	6	7
15	2	3	10	8	9	4

- 2. Que fait cet algorithme?
- 3. Quelle est sa complexité dans le pire des cas? Dans le meilleur des cas?

Exercice 2 Yaourts (4 points)

La production de yaourt consiste à introduire des bactéries (Streptococcus thermophilus & Lactobacillus bulgaricus) dans un volume de lait. En conditions favorables (température de 43 degrés et présence de sucres), les populations bactériennes connaissent une croissance dite exponentielle si le milieu n'est pas limitant (i.e. présence de sucres suffisante). Le taux de croissance est alors constant. On considère un taux de croissance horaire μ égal à 2.5 pour les populations qui nous intéressent (toutes les heures la population est augmentée de 2,5% de sa valeur).

1.	Ecrire une fonction prenant en paramètre le nombre de bactéries initial et qui calcule et donne au bout de combien d'heures la population aura doublée, en supposant que la mortalité est nulle.
2.	Une fois stocké au froid, la croissance bactérienne s'arrête. Alors, d'un point de vue pratique, chaque heure, 10% des bactéries meurent. Écrire une fonction qui calcule et retourne au bout de combien de temps (heures) la population bactérienne totale comporte au moins 20% de bactéries mortes pour une quantité de bactéries initiale donnée. (La réponse n'est pas 2!)

Structure de Données A3

Exercice 3 Structure de tas (6 points)

Un tas ascendant est un arbre binaire vérifiant les propriétés suivantes :

— la différence maximale de profondeur entre deux feuilles est de 1 (*i.e.* toutes les feuilles se trouvent sur la dernière ou sur l'avant-dernière ligne);

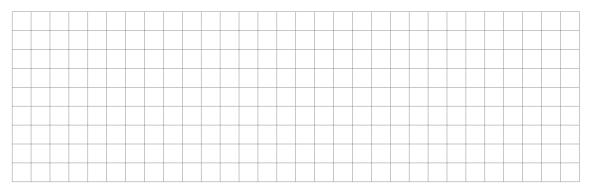
- les feuilles de profondeur maximale sont "tassées" sur la gauche.
- chaque nœud est de valeur supérieure à celle de ces deux fils.

Un tas ou un arbre binaire presque complet peut être **stocké dans un tableau**, en posant que les deux descendants de l'élément d'indice i sont les éléments d'indices 2i et 2i+1 (pour un tableau indicé à partir de 1). En d'autres termes, les nœuds de l'arbre sont placés dans le tableau ligne par ligne, chaque ligne étant décrite de gauche à droite.

L'insertion d'un élément dans un tas se fait de la façon suivante : on place l'élément sur la première case libre et on échange l'élément et son père quand ce dernier est inférieur et qu'il existe.

Quand la racine est modifiée, l'opération de **tamisage** consiste à échanger la racine avec le plus grand de ses fils, et ainsi de suite récursivement jusqu'à ce qu'elle soit à sa place.

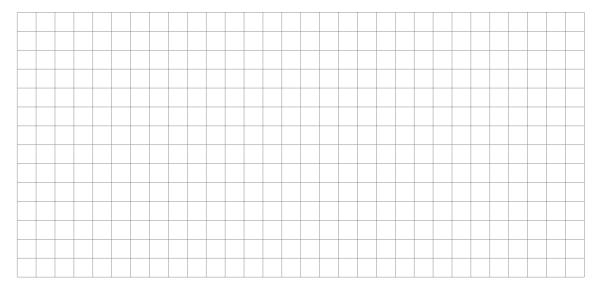
1. Dessinez un tas correspondant au tableau T: [14, 10, 8, 4, 9, 2, 7, 1]



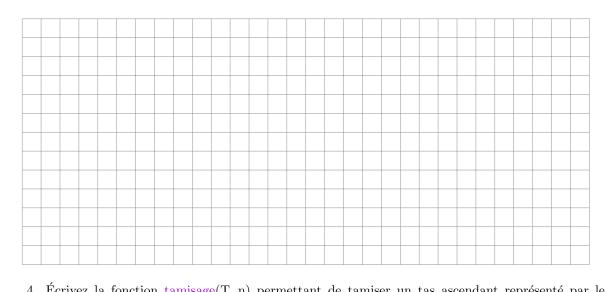
2. Insérez la valeur 12 dans le tas et donnez le tas final.



3. Remplacez la racine du tas par la valeur 3 et détaillez le tamisage.



Structure de Données A4



Structure de Données A5

Exercice 4 Queue de priorité (6 points)

En informatique, une queue de priorité est un type abstrait élémentaire sur laquelle on peut effectuer quatre opérations :

- 1. insérer un élément représenté par sa clé de priorité (insérer(x, Q, n));
- 2. lire (puis supprimer) l'élément ayant la plus grande clé;
- 3. tester si la queue de priorité est vide ou pas (estVide(Q));
- 4. augmenter la clé d'un élément.

Pour gérer ces opérations on utilisera un tas ascendant (représenté par un tableau). Les opérations disponibles sur les tas sont :

- l'insertion avec insertion (x, T, n) qui insère l'élément x dans le tas ascendant représenté par le tableau T contenant n éléments;
- et le tamisage avec tamisage(T, n) qui tamise un tas ascendant représenté par le tableau T contenant n éléments.

1.	Écrivez la fonction $lire(Q, n)$ qui retourne et supprime l'élément ayant la plus grande clé de la queue de priorité Q dont la taille est n . Attention Q doit toujours correspondre à un tas ascendant après la suppression.
2.	Écrivez la fonction plusGrandCommun(Q1, n1, Q2, n2) qui retourne l'élément ayant la plus grande clé en commun dans les deux queues de priorité Q1 et Q2, ou 0 si les deux de queues de priorité n'ont aucun éléments en commun. Q1 et Q2 possèdent respectivement n1 et n2 éléments. Remarque dans cette fonction, Q1 et Q2 peuvent être modifiés.

Structure de Données	A6