EXAMEN INF301 2022
Algorithmique & Prog. Impérative
13 janvier 2023

Durée : 2h
Tous documents interdits
Une feuille A4 R/V manuscrite autorisée

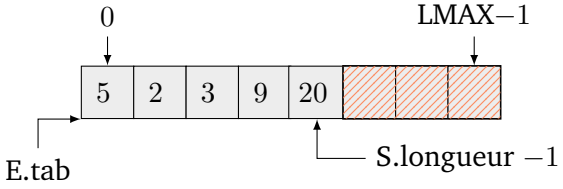
Sauf indication contraire, les Questions à Choix ont une **unique bonne réponse**.
Les questions faisant apparaître le symbole ♣ peuvent présenter zéro, une ou plusieurs bonnes réponses.

Les réponses aux Questions à Choix sont à donner exclusivement sur la feuille de réponse (page 5 à détacher). Les réponses aux questions ouvertes sont à donner sur votre copie d'examen.

Cet examen comporte 20 points répartis sur 3 exercices. Le barème est indicatif et pourra être modifié.

Exercice 1 (Séquences et tableaux (5 points))

On considère la séquence S à droite pour tout l'exercice. Chaque question est indépendante et **ne modifie pas S pour les autres questions**.



Question 1 [1 pt]

Que s'afficherait-il à l'exécution du code suivant :

```
affiche (S.longueur, ", ")
ajoute_fin (S, 5)
affiche (S.longueur)
```

- A « 20, 6 »
- B « 5, 5 »
- C « 20, 5 »
- D « 5, 6 »
- E *Aucune réponse correcte.*
- F *Toutes les réponses sont correctes.*
- G *Manque de données dans l'énoncé.*
- H *La question est absurde.*

Question 2 [1 pt]

Si on exécute « affiche(S) », on obtiendrait la chaine suivante : « 5, 2, 3, 9, 20 ». Qu'obtiendrait-on si l'on exécutait ceci :

```
S.longueur ← S.longueur+1
affiche(S)
```

- A Erreur mémoire
- B « 5, 2, 3, 9, 20, 5 »
- C « 5, 2, 3, 9, 20 »
- D « 5, 2, 3, 9, 20, 1 »
- E *Aucune réponse correcte.*
- F *Toutes les réponses sont correctes.*
- G *Manque de données dans l'énoncé.*
- H *La question est absurde.*

On veut déplacer le dernier élément de S en première position. Complétez les trous de l'algorithme proposé.

```
A:
pour chaque i de B: faire
    S.tab[i+1] ← S.tab[i]
C:
```

Question 3 Que doit contenir la boîte A ? [0.5 pt]

- A x ← S.tab[S.longueur]
- B tmp ← S.tab[S.longueur−1]
- C x ← S.tab[S.longueur−1]
- D tmp ← S.tab[S.longueur]
- E *Aucune réponse correcte.*
- F *Toutes les réponses sont correctes.*
- G *Manque de données dans l'énoncé.*
- H *La question est absurde.*

Question 4 Que doit contenir la boîte B ? [1 pt]

- A S.longueur−1 à 1
- B 1 à S.longueur−2
- C S.longueur−2 à 0
- D 0 à S.longueur−1
- E *Aucune réponse correcte.*
- F *Toutes les réponses sont correctes.*
- G *Manque de données dans l'énoncé.*
- H *La question est absurde.*

Question 5 Que doit contenir la boîte C ? [0.5 pt]

- A S.tab[tmp] ← x
- B S.tab[0] ← tmp
- C S.tab[x] ← 0
- D S.tab[tmp] ← 0
- *E* *Aucune réponse correcte.*
- *F* *Toutes les réponses sont correctes.*
- *G* *Manque de données dans l'énoncé.*
- *H* *La question est absurde.*

Question 6 [1 pt]

Que s'afficherait-il à l'exécution du code suivant :

```
supprime_elt (S, 20)
affiche (S.tab[S.longueur])
```

- A 9
- B Erreur mémoire
- C 20
- D Nil
- *E* *Aucune réponse correcte.*
- *F* *Toutes les réponses sont correctes.*
- *G* *Manque de données dans l'énoncé.*
- *H* *La question est absurde.*

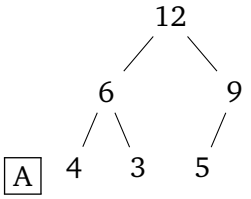
Exercice 2 (Tri par tas (10 points))

Dans cet exercice, on cherche à construire une séquence triée dans l'ordre croissant de toutes les valeurs contenues dans un arbre binaire. La séquence sera une liste chaînée. L'arbre possède de plus la propriété de **tas**. Un tas est un arbre binaire qui a la propriété suivante : pour chaque nœud n, sa valeur n.val est toujours plus grande ou égale à **toutes les valeurs des nœuds des deux sous-arbres** de n (gauche et droit).
Nous utilisons les types et structures présentés à droite.

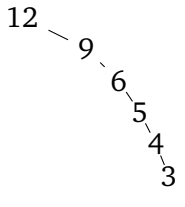
```
type tas : arbre
type arbre : référence vers nœud
type nœud { val : entier
            gauche : arbre
            droit : arbre
}
```

Question 7 ♣ Parmi les arbres suivants, lesquels sont des tas ? [1 pt]

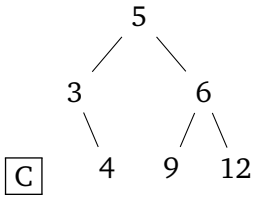
A



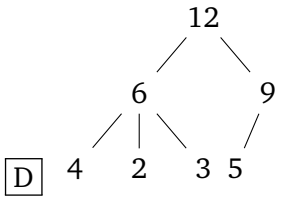
B



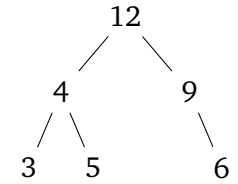
C



D



E



F Aucun n'est un tas

Question 8 [1 pt]

On cherche un algorithme `vérifie_tas` qui renvoie Vrai si un arbre est un tas et Faux sinon. On a déjà la fonction `vérifie_nœud` ci à droite. Comment `vérifie_tas` doit-elle appeler `vérifie_nœud` pour être correcte ?

```
vérifie_nœud(n, v)
    si n.val > v alors retourner Faux
    retourner vérifie_nœud(n.gauche, n.val)
              et vérifie_nœud(n.droit, n.val)
```

A vérifie_nœud(A, A.val)
B vérifie_nœud(A, 0)
C vérifie_nœud(A.gauche, A.val) et vérifie_nœud(A.droit, A.val)
D vérifie_nœud(A, $+\infty$)
E Aucune réponse correcte.
F Toutes les réponses sont correctes.
G Manque de données dans l'énoncé.
H La question est absurde.

Question 9 On veut extraire la valeur maximale d'un tas. Écrire une fonction `extrait_max`(T) qui retourne la valeur maximale d'un tas T tout en modifiant T de manière à ce que la valeur maximale soit supprimée du tas. T doit garder la propriété de tas.
(La fonction doit être récursive ou utiliser une ou plusieurs fonctions auxiliaires récursives.)
[3 pts]

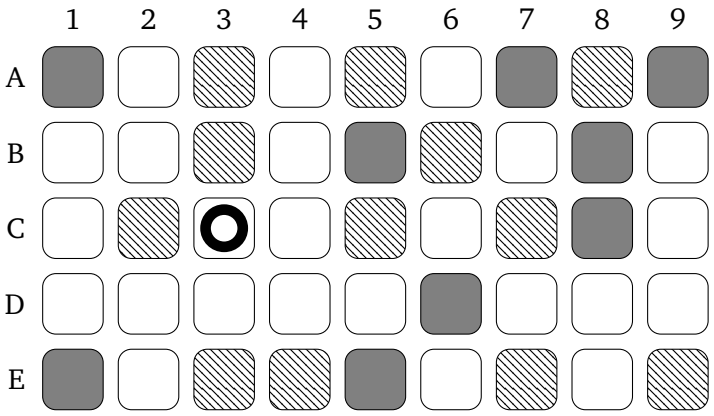
Question 10 Donnez et justifiez la complexité de votre fonction `extrait_max`. [1 pt]

Question 11 Utilisez la (ou les) fonction(s) précédente(s) pour écrire un algorithme qui prend un tas en entrée, et retourne une liste chaînée contenant les valeurs de l'arbre dans l'ordre croissant. Vous avez le droit de modifier le tas donné en entrée. [3 pts]

Question 12 Donnez (et justifiez) la complexité de votre algorithme de tri. [1 pt]

Exercice 3 (Exécution d'algorithmes (4 points))

On considère l'environnement à droite dans lequel se déplace le rond situé initialement en C3. Le but est de retrouver pour plusieurs programmes la case d'arrivée de ce rond. Par exemple, après le programme « `haut(); droite()` », le résultat serait B4. Les conditions portent sur l'état de la case où se situe le rond. Par exemple, le programme « `droite(); si` ● `droite()` » aurait pour résultat C4.



Pour ces questions, cochez exactement deux cases parmi les réponses proposées : une lettre et un nombre, ou bien la case « Autre » si rien ne correspond (par exemple si le rond sort de l'environnement).
(Note : les programmes commencent par le « main » et sont indépendants : la position de départ est toujours C3.)

Question 13 ♣

main :
- x ← 1
- **tant que** ≠● **faire**
  - haut()
  - **faire** x **fois**
    - droite()
  - x ← x+1
  - bas()
  - **faire** x **fois**
    - gauche()

Question 14 ♣

Fonction *f()*
- bas()
- **tant que** ○ **faire**
  - droite()
  - f()
  - gauche()
- droite()

main :
- f()

Question 15 ♣

Fonction *g(x)*
- **si** x = 2 **alors**
  - **retourner**
- **faire** x **fois**
  - droite()
- **si** ● **alors**
  - x ← x+1
- haut()
- g(x)
- **faire** x **fois**
  - gauche()

main :
- g(1)

Question 16 ♣

Fonction *h()*
- **si** ○ **alors**
  - droite()
  - haut()
  - h()
- **si** ▨ **alors**
  - droite()
  - bas()

main :
- h()

Feuille de réponses

Noircissez <u>entièrement</u> les cases.

Les réponses aux QCM sont à donner exclusivement sur cette feuille : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

⟵ codez votre **<u>numéro d'anonymat</u>** ci-contre, et **<u>ré-inscrivez le</u>** ci-dessous.

↓

Numéro d'anonymat :

.....

Question 1 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 2 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 3 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 4 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 5 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 6 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 7 : ☐A ☐B ☐C ☐D ☐E ☐F

Question 8 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 9 : ..... Extrait ☐W ☐II ☐I ☐P ☐PP ☐C *Réservé*

Question 10 : ..... Cpx ☐W ☐I ☐P ☐C *Réservé*

Question 11 : ..... Tri ☐W ☐II ☐I ☐P ☐PP ☐C *Réservé*

Question 12 : ..... Tri Cpx ☐W ☐I ☐P ☐C *Réservé*

Question 13 : ☐C ☐9 ☐3 ☐7 ☐B ☐5 ☐4 ☐6 ☐2 ☐E ☐A ☐8 ☐1 ☐D ☐Autre

Question 14 : ☐9 ☐1 ☐B ☐6 ☐4 ☐D ☐A ☐2 ☐7 ☐3 ☐8 ☐E ☐C ☐5 ☐Autre

Question 15 : ☐B ☐8 ☐D ☐6 ☐A ☐7 ☐1 ☐9 ☐4 ☐2 ☐E ☐3 ☐C ☐5 ☐Autre

Question 16 : ☐1 ☐D ☐C ☐9 ☐8 ☐7 ☐E ☐2 ☐5 ☐A ☐B ☐3 ☐4 ☐6 ☐Autre