Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour consulter tous nos sujets d'annales et corrigés du Bac Général Maths + NSI, veuillez visiter :
bac-maths-nsi.fr/annales-sujets-et-corriges

Polynésie - mai 2022 (corrigé)

Exercice 1 (Programmation et récursivité)

1. (a) La fonction `A` s'appelle elle-même, elle est donc récursive.
   (b) Si on obtient `False` un très grand nombre de fois consécutives, il y aura trop d'appels récursifs et cela provoquera l'erreur `Recursion error` car la taille de la pile d'exécution sera dépassée.
2. (a) Le code suivant convient

```
def A(n):
    if n <= 0 or choice([True,False]):
        return "a"
    else:
        return "a" + A(n-1) + "a"
```

   (b) L'appel `A(50)` termine toujours :
   - soit `choice([True,False])` n'a jamais renvoyé `True` et dans ce cas il y a eu 50 appels récursifs jusqu'à ce que `n` vale 0 et exécute le cas de base en renvoyant `"a"` ;
   - soit `choice([True,False])` a renvoyé `True` avant que `n` atteigne la valeur 0.

3. - `B(0)` renvoie `"bab"`.
   - `B(1)` renvoie `"bab"` ou `"bbabb"`.
   - `B(2)` renvoie `"bab"` ou `"baaab"` ou `"bbabb"` ou `"bbbabbb"`
4. (a) Le code suivant convient :

```
def regleA(chaine):
    n = len(chaine)
    if n >= 2:
        return chaine[0] == "a" and choice[n-1] == "a" and
            regleA(raccourcir(chaine))
    else:
        return chaine == "a"
```

   (b) Le code suivant convient :

```
def regleB(chaine):
    n = len(chaine)
    if n >= 2:
        return chaine[0] == "b" and choice[n-1] == "b" and
            (regleA(raccourcir(chaine)) or regleB(raccourcir(chaine)))
    else:
        return False
```

Exercice 2 (Architecture matérielle, ordonnancement et expressions booléennes)

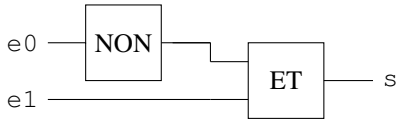
1. Voici l'annexe 1 complétée :

Numéro de périphérique	Adresse	Opération	Réponse de l'ordonnanceur
0	10	écriture	OK
1	11	lecture	OK
2	10	lecture	ATT
3	10	écriture	ATT
0	12	lecture	OK
1	10	lecture	OK
2	10	lecture	OK
3	10	écriture	ATT

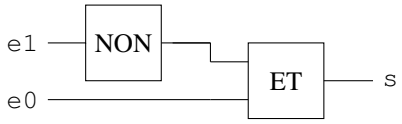
2. Le périphérique 1 obtient toujours `ATT` comme réponse de l'ordonnanceur car le périphérique 0 utilise déjà l'adresse 10 en écriture.
3. (a) **Au tour 1** le périphérique 0 peut écrire mais le périphérique 1 ne peut pas lire.
   Au tour 2 le périphérique 0 ne peut pas écrire mais le périphérique 1 peut lire.
   Au tour 3 le périphérique 0 peut écrire mais le périphérique 1 ne peut pas lire.
   Au tour 4 le périphérique 0 peut écrire mais le périphérique 1 ne peut pas lire.
   (b) Le périphérique 1 lit effectivement une fois sur trois ce qui a été écrit par le périphérique 0 à l'adresse 10.
4. Voici l'annexe 2 complétée :

Tour	Numéro du périphérique	Adresse	Opération	Réponse de l'ordonnanceur	ATT_L	ATT_E
1	0	10	écriture	OK	vide	vide
1	1	10	lecture	ATT	(1,10)	vide
1	2	11	écriture	OK	(1,10)	vide
1	3	11	lecture	ATT	(1,10), (3,11)	vide
2	1	10	lecture	OK	(3,11)	vide
2	3	11	lecture	OK	vide	vide
2	0	10	écriture	ATT	vide	(0,10)
2	2	12	écriture	OK	vide	(0,10)
3	0	10	écriture	OK	vide	vide
3	1	10	lecture	ATT	(1,10)	vide
3	2	11	écriture	OK	(1,10)	vide
3	3	12	lecture	OK	(1,10)	vide

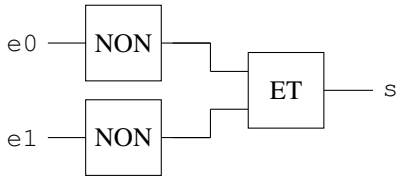
5. (a) On complète le schéma de la façon suivante :



   (b) Le schéma suivant convient :



   (c) Le schéma suivant convient :



Exercice 3 (Base de données, modèle relationnel et langage SQL)

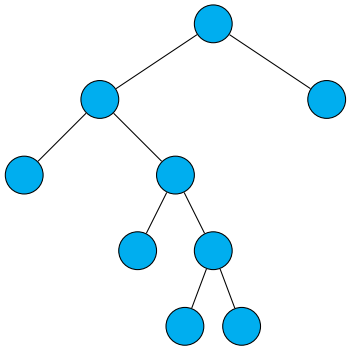
1. (a) `SELECT ip, nompage FROM Visites;`
   (b) `SELECT DISTINCT ip FROM Visites;`
   (c) `SELECT nompage FROM Visites WHERE ip = "192.168.1.91";`
2. (a) L'attribut `identifiant` est la clé primaire de la table `Visites`.
   (b) L'attribut `identifiant` est la clé étrangère de la table `Pings`.
   (c) Le système de gestion de base de données va vérifier avant chaque ajout dans la table `Pings` que l'atttribut `identifiant` existe bien dans la table `Visites`.
3. `INSERT INTO Pings VALUES (1534,105);`
4. (a) `UPDATE Pings SET duree = 120 WHERE identifiant = 1534;`
   (b) Il se peut que pendant le routage les requêtes n'empruntent pas le même chemin et arrivent dans un ordre différent.
   (c) On peut imaginer que la réception d'un doublet `(1534,120)` arrive après celle du doublet `(1534,135)` par exemple ce qui pose un problème dans le cas d'une mise à jour car la durée `120` serait enregistrée à la place de `135`. En ne réalisant que des insertions on évite ce problème, par contre il faudra chercher la valeur maximum de durée associée à un identifiant dans la table `Pings` si on veut connaître la durée de connexion.
5. ```
   SELECT Visites.nompage FROM Visites
   JOIN Pings ON Pings.identifiant = Visites.identifiant
   WHERE Pings.duree > 60;
   ```
   ```

Exercice 4 (Structure de données, piles)

1. On complète la ligne 6 ainsi : `if e1 > e2:` et la ligne 8 avec : `e1 = e2`
2. (a) La valeur renvoyée par l'appel `A.est_triee()` renvoie `False` car `4` est plus grand que `3`.
   (b) Après l'exécution de cette instruction, `A` vaut `[1,2]` (les élements `3` et `4` ont été dépilés pour comparaison).
3. On complète la ligne 9 ainsi : `maxi = elt` et la ligne 11 avec : `q.empiler(elt)`
4. (a) **Initialisation** `B` vaut `[9, -7, 8, 12]` et `q` vaut `[]`
   Itération 1 `B` vaut `[9, -7, 8]` et `q` vaut `[4]`
   Itération 2 `B` vaut `[9, -7]` et `q` vaut `[4, 8]`
   Itération 3 `B` vaut `[9]` et `q` vaut `[4, 8, -7]`
   Itération 4 `B` vaut `[]` et `q` vaut `[4, 8, -7, 9]`
   (b) Avant l'exécution de la ligne 14, la pile `q` est vide, elle vaut `[]` et la pile `B` contient la liste `[9, -7, 8, 4]`.
   (c) Si `B = [12, 5, 10]` alors le résultat obtenu est `[10,5]`, l'ordre des éléments est modifié.
5. (a) **Avant la ligne 3** `B` vaut `[1, 6, 4, 3, 7, 2]` et `q` vaut `[]`
   Avant la ligne 5 `B` vaut `[]` et `q` vaut `[7, 6, 4, 3, 2, 1]`
   À la fin `B` vaut `[1, 2, 3, 4, 5, 6, 7]` et `q` vaut `[]`
   (b) Les données de la liste représentant la pile `B` sont dans l'ordre croissant donc la pile contient les éléments dans l'ordre décroissant. La fonction `traiter` trie les données.
   En effet, la boucle `while` à la ligne 3 permet d'empiler dans la pile `q` les éléments de la pile `B` du plus grand au plus petit et la boucle `while` de la ligne 5 permet d'empiler de nouveau tous les éléments de `q` dans la pile `B`, ce qui a pour effet d'empiler le plus petit en premier et le plus grand en dernier.

Exercice 5 (Algorithmique et arbres binaires)

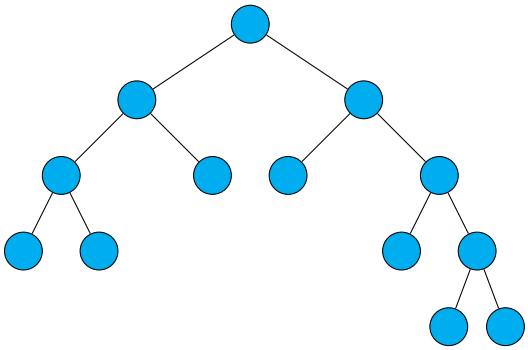
1. (a) L'arbre a pour hauteur 2. En effet, le sous-arbre droit est réduit à la racine, il a donc pour hauteur 0, mais le sous-arbre-gauche est constitué d'une racine avec un fils gauche (une feuille), il est donc de hauteur 1.
   (b) Voici un arbre de hauteur 4 (il a plusieurs possibilités) :



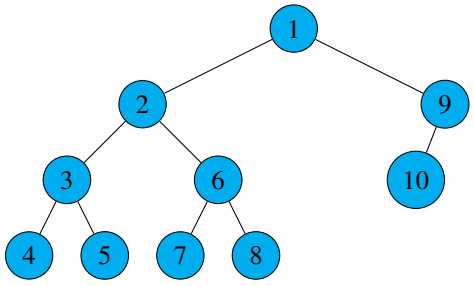
2. On peut compléter les lignes 7 et 8 de la façon suivante :

```
sinon, si sous_arbre_droit(A) vide:
    renvoyer 1 + hauteur(sous_arbre_gauche(A))
```

3. (a) Puisque l'arbre `R` a pour hauteur 4 et non 3 (la hauteur de son sous-arbre gauche augmentée de 1), cela signifie que son sous-arbre droit a pour hauteur 3, il n'est donc pas vide.
   (b) Voici un arbre possible :



4. (a) Dans le cas de l'arbre binaire proposé à la question 1.a nous avons $h = 2$ et $n = 4$.
   Ainsi $h + 1 = 3$ et $2^{h+1} - 1 = 2^3 + 1 = 8 - 1 = 7$ et on a bien $3 \leqslant 4 \leqslant 7$ donc l'inégalité est vérifiée.
   (b) Pour obtenir un arbre de hauteur h constitué de $h + 1$ nœuds, il suffit de construire un arbre dont chaque nœud interne a exactement un fils (arbre filiforme).
   (c) Pour obtenir un arbre de hauteur h constitué de $2^{h+1} - 1$ nœuds, il faut construire un arbre complet (tous les nœuds internes ont exactement deux fils).
5. Voici la numérotation obtenue avec un parcours préfixe (racine-gauche-droit).



6. On peut compléter la ligne 7 de la façon suivante : `return arbre(arbre_vide(), arbre_vide()`
   Et la ligne 11 ainsi : `droite = annexe(hauteur_max - 1`.

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.