Math93.com

BAC NSI - Correction Centres Étrangers G2 (Liban) 2023 - Sujet 1 - NSI

Thèmes des exercices (sur 12 points)

- Exercice 1 (3 points) : Algorithmique et la programmation (fonction récursive).
- Exercice 2 (5 points) : Bases de données, la représentation des données et les réseaux.
- Exercice 3 (4 points) : Piles, des arbres et de l'algorithmique.
- Lien vers le sujet

Exercice 1. Algorithmique et la programmation (fonction récursive). 3 points

Un palindrome est un mot qui se lit de la même manière de la gauche vers la droite que de la droite vers la gauche (exemple : « kayak » est un palindrome). On propose ci-dessous une fonction pour tester si un mot est un palindrome. On précise que, pour une chaîne de caractères chaine :

- l'instruction len(chaine) renvoie sa longueur ;
- l'instruction chaine[-1] renvoie son dernier caractère ;
- l'instruction chaine[1 : -1] renvoie la chaîne privée de son premier caractère et de son dernier caractère.

```
1 def tester_palindrome(chaine):
2     if len(chaine) < 2:
3         return True
4     elif chaine[0] != chaine[-1]:
5         return False
6     else:
7         chaine = chaine[1:-1]
8         return tester_palindrome(chaine)
```

1. On saisit, dans la console, l'instruction suivante : tester_palindrome('kayak') Combien de fois est appelée la fonction tester_palindrome lors de l'exécution de cette instruction ? On veillera à compter l'appel initial.

Corrigé

En exécutant l'instruction tester_palindrome('kayak') , la fonction sera appelée 3 fois.

- L'appel initial teste la chaîne "kayak".
- Le deuxième appel teste la sous-chaîne "aya".
- Le troisième appel teste la sous-chaîne "y".

À partir de là, la fonction commence à remonter la pile d'appels récursifs, en renvoyant True pour la sous-chaîne "y" car de longueur <2, puis True pour la sous-chaîne "aya", et enfin True pour la chaîne "kayak" initiale.

2.

(a) Justifier que la fonction tester_palindrome est récursive.

Corrigé

La fonction tester_palindrome est récursive car elle s'appelle elle-même dans son corps de fonction. Elle se divise en sous-problèmes en testant si les caractères à chaque extrémité de la chaîne sont les mêmes et en répétant le processus pour la sous-chaîne interne. Cette division en sous-problèmes se poursuit jusqu'à ce qu'il ne reste que moins de 2 caractères à traiter.

(b) Expliquer pourquoi l'appel à la fonction tester_palindrome se terminera quelle que soit la chaîne de caractères sur laquelle elle s'applique.

Corrigé

L'appel à la fonction tester_palindrome se terminera toujours car à chaque itération de l'appel récursif, la taille de la chaîne est réduite d'au moins deux caractères, jusqu'à ce qu'elle atteigne une longueur de 0 ou 1. À ce stade, la fonction retourne True puisque la condition ln(chaine) < 2 n'est plus vérifiée.

3. La saisie, dans la console, de l'instruction tester_palindrome(53235) génère une erreur.

(a) Parmi les quatre propositions suivantes, indiquer le type d'erreur affiché :

- ZeroDivisionError
- ValueError
- TypeError
- IndexError

Corrigé

La saisie de l'instruction tester_palindrome(53235) génère une erreur car 53235 n'est pas une chaîne de caractères mais un entier.
Le premier test if len(chaine) < 2 : va renvoyer une erreur de type :

TypeError : object of type 'int' has no len()

(b) Proposer sur la copie une ou plusieurs instructions qu'on pourrait écrire entre la ligne 1 et la ligne 2 du code de la fonction tester_palindrome et permettant d'afficher clairement cette erreur à l'utilisateur.

Corrigé

Il faut mettre un assert.

```
assert type(chaine) == str,"l'argument doit étre de type string(str)"
```

4. Écrire le code d'une fonction itérative (non récursive) est_palindrome qui prend en paramètre une chaîne de caractères et renvoie un booléen égal à True si la chaîne de caractères est un palindrome, False sinon.

Corrigé

Voici une proposition :

```
def est_palindrome(chaine):
    chaine=str(chaine)
    n=len(chaine)
    for i in range(n//2):
       if chaine[i]!=chaine[n-i-1]:
           return False
    return True
```

ou

```
def est_palindrome(chaine):
    while len(chaine) >= 2 :
        if chaine[0] != chaine[-1]:
            return False
        chaine = chaine[1:-1]
    return True
```

Exercice 2. Bases de données, la représentation des données et les réseaux. 5 points

Cet exercice utilise certains des mots-clés du langage SQL suivants : DELETE, FROM, INSERT, INTO, JOIN, ON, SELECT, SET, UPDATE, VALUES, WHERE .

Les vacances d'été se rapprochent et le propriétaire d'une pension pour animaux gère les places dont il dispose à l'aide d'une base de données dont voici le schéma relationnel :

- client(num_client, nom_client, prenom_client, mail_client, tel_client)
- animal(num_animal, nom_animal, categorie_animal, taille_animal, num_client)
- cage(num_cage, taille_cage, secteur_cage)
- reservation(num_reservation, date_debut_reservation, date_fin_reservation, num_client, num_animal, num_cage)

Ci-dessous, on donne des extraits des tables client, animal, cage et reservation .

Extrait de la table `client` :

num_client	nom_client	prenom_client	mail_client	tel_client
16	Dupont	Marc	marc.dupont@mail.com	0604050401
345	Morel	Fabien	fabien.morel@mail.com	0700051020

Extrait de la table `animal` :

num_animal	nom_animal	categorie_animal	taille_animal	num_client
22	Yuki	souris	petit	16
112	Balou	chat	moyen	141
320	Api	chien	grand	237
423	Rex	chien	moyen	259
491	Rex	chien	petit	345

Extrait de la table `cage` :

num_cage	taille_cage	secteur_cage
4	grand	chien
12	petit	chien
23	moyen	chien
31	moyen	chien
32	petit	rongeur
33	grand	chat

Extrait de la table `reservation` :

num_reservation	date_debut_reservation	date_fin_reservation	num_client	num_animal	num_cage
44	2022-08-23	2022-08-25	26	12	12
45	2022-07-11	2022-07-22	345	491	23
46	2022-08-11	2022-08-22	345	491	23
47	2022-08-23	2022-09-10	345	491	23
48	2022-10-11	2022-10-22	345	491	23

1. Étude du schéma relationnel

(a) Pour chaque attribut de la relation cage, spécifier son type, en utilisant le tableau des types suivant :

CHAR(t) VARCHAR(t)	Texte de longueur fixe de t caractères. Texte de longueur variable de t caractères au maximum.
INT	Nombre entier de -2^{31} à $2^{31}-1$ (signé) ou de 0 à $2^{32}-1$ (non signé).
FLOAT	Réel à virgule flottante.
DATE DATETIME	Date format AAAA-MM-JJ. Date et heure format AAAA-MM-JJ HH :MI :SS.

Corrigé

On obtient :

cage(num_cage : INT , taille_cage : VARCHAR(6) , secteur_cage : VARCHAR(10))

(b) Préciser, pour la relation reservation, le nom de la clé primaire pouvant être utilisée.

Corrigé

reservation(num_reservation, date_debut_reservation, date_fin_reservation, num_client, num_animal, num_cage)

Extrait de la table `reservation` :

num_reservation	date_debut_reservation	date_fin_reservation	num_client	num_animal	num_cage
44	2022-08-23	2022-08-25	26	12	12
45	2022-07-11	2022-07-22	345	491	23
46	2022-08-11	2022-08-22	345	491	23
47	2022-08-23	2022-09-10	345	491	23
48	2022-10-11	2022-10-22	345	491	23

La clé primaire de la relation reservation pourrait être le champ num_reservation qui doit être unique pour chaque réservation.

(c) Indiquer, pour la relation reservation, la ou les clés étrangères (ou secondaires) et en indiquer l'utilité.

Corrigé

La relation reservation contient trois clés étrangères :

- num_client : clé étrangère faisant référence à la relation client, indiquant le client qui a réservé la cage pour son animal.
- num_animal : clé étrangère faisant référence à la relation animal, indiquant l'animal pour lequel la cage a été réservée.
- num_cage : clé étrangère faisant référence à la relation cage, indiquant la cage qui a été réservée.

L'utilité de ces clés étrangères est de lier les informations de la table reservation aux

- informations correspondantes dans les tables client, animal et cage.

2. Requêtes

   (a) Indiquer le résultat de l'exécution de la requête suivante :

```
SELECT nom_animal
FROM animal
WHERE categorie_animal = 'chien';
```

Corrigé

Extrait de la table `animal` :

num_animal	nom_animal	categorie_animal	taille_animal	num_client
22	Yuki	souris	petit	16
112	Balou	chat	moyen	141
320	Api	chien	grand	237
423	Rex	chien	moyen	259
491	Rex	chien	petit	345

Cette requête permet de sélectionner tous les noms des animaux qui appartiennent à la catégorie "chien" dans la table "animal". Le résultat sera une liste de noms de chiens soit ici :

nom_animal
Api
Rex
Rex

(b) Écrire une requête SQL permettant d'afficher les noms de tous les clients dont l'animal a occupé la cage numéro 23.

Corrigé

Pour afficher les noms de tous les clients dont l'animal a occupé la cage numéro 23 :

- On sélectionne le nom du client dans la table "client"
- On utilise les jointures INNER JOIN pour lier les tables "client", "animal", "reservation" et "cage"
- On relie les animaux à leur propriétaire grâce à la clé étrangère "num_client"
- On relie les réservations aux animaux grâce à la clé étrangère "num_animal"
- On relie les cages aux réservations grâce à la clé étrangère "num_cage"
- On filtre les résultats en ne gardant que ceux où le numéro de cage est égal à 23.

Extrait de la table `client` :

num_client	nom_client	prenom_client	mail_client	tel_client
16	Dupont	Marc	marc.dupont@mail.com	0604050401
345	Morel	Fabien	fabien.morel@mail.com	0700051020

Extrait de la table `animal` :

num_animal	nom_animal	categorie_animal	taille_animal	num_client
22	Yuki	souris	petit	16
112	Balou	chat	moyen	141
320	Api	chien	grand	237
423	Rex	chien	moyen	259
491	Rex	chien	petit	345

Extrait de la table `cage` :

num_cage	taille_cage	secteur_cage
4	grand	chien
12	petit	chien
23	moyen	chien
31	moyen	chien
32	petit	rongeur
33	grand	chat

Extrait de la table `reservation` :

num_reservation	date_debut_reservation	date_fin_reservation	num_client	num_animal	num_cage
44	2022-08-23	2022-08-25	26	12	12
45	2022-07-11	2022-07-22	345	491	23
46	2022-08-11	2022-08-22	345	491	23
47	2022-08-23	2022-09-10	345	491	23
48	2022-10-11	2022-10-22	345	491	23

```sql
SELECT client.nom_client
FROM client
INNER JOIN reservation ON reservation.num_client = client.num_client
WHERE reservation.num_cage = 23;
```

(c) Un nouvel animal doit être enregistré dans la base de données qui contient actuellement 491 animaux. Il s'appelle Suki, c'est un chat de petite taille dont le propriétaire a déjà été enregistré sous le numéro 342. Écrire la requête SQL permettant d'insérer ces nouvelles données dans la base de données.

Corrigé

Voici la table animal dans laquelle on va devoir insérer Suki :

Extrait de la table `animal` :

num_animal	nom_animal	categorie_animal	taille_animal	num_client
22	Yuki	souris	petit	16
112	Balou	chat	moyen	141
320	Api	chien	grand	237
423	Rex	chien	moyen	259
491	Rex	chien	petit	345

La requête sera :

```
INSERT INTO animal
VALUES (492, 'Suki', 'chat', 'petite', 342);
```

3. **Programmation Python**

Suite à une panne, le responsable de la pension n'a plus accès à sa base de données. Heureusement, il avait fait une sauvegarde de ses tables au format csv. Il les a importées à l'aide d'un programme Python, chacune sous la forme d'une liste de dictionnaires. Pour simplifier, on considérera que la table reservation est la liste de dictionnaires suivante :

```
reservation = [
{'num_reservation' : 44, 'date_debut_reservation' : '2022-08-23',
'date_fin_reservation' : '2022-08-25', 'num_client' : 26,
'num_animal' : 12, 'num_cage' : 12},
{'num_reservation' : 45,'date_debut_reservation' : '2022-07-11',
'date_fin_reservation' :  '2022-07-22', 'num_client' : 345,
'num_animal' : 491, 'num_cage' : 23},
{'num_reservation' : 46,'date_debut_reservation' : '2022-08-11',
'date_fin_reservation' : '2022-08-22', 'num_client' : 345,
'num_animal' : 491, 'num_cage' : 23},
{'num_reservation' : 47, 'date_debut_reservation' :  '2022-08-23',
'date_fin_reservation' : '2022-09-10',  'num_client' : 345,
'num_animal' : 491,'num_cage' : 23},
{'num_reservation' : 48, 'date_debut_reservation' : '2022-10-11',
'date_fin_reservation' : '2022-10-22', 'num_client' : 345,
'num_animal' : 491,'num_cage' : 23}]
```

(a) On donne ci-dessous, le code Python d'une fonction mystere.

```
def mystere(table,date):
    liste=[]
    for ligne in table:
        if ligne['date_debut_reservation'] == date:
            liste.append(ligne['num_client'])
    return liste
```

On rappelle que l'appel L.append(x) ajoute l'élément x à la fin de la liste L. Indiquer l'affichage produit par l'exécution de la ligne de code suivante :

```
print(mystere(reservation, '2022-08-23'))
```

Corrigé

La fonction mystere(table, date) prend en entrée une table (ici la table "reservation" sous forme de liste de dictionnaires) et une date au format string ('YYYY-MM-DD') et renvoie une liste des numéros de clients ayant effectué une réservation débutant à cette date.

L'exécution de la ligne de code print(mystere(reservation, '2022-08-23')) renverra la liste [26, 345], car deux réservations ont été effectuées le 23 août 2022, l'une par le client 26 et l'autre par le client 345.

(b) Le responsable de la pension veut obtenir le nombre de réservations qui ont été effectuées pour un numéro de client donné.
Écrire les lignes de code après la ligne 7 de la fonction nombre_reservation afin de respecter la spécification donnée ci-dessous.

```
def nombre_reservation(table,numero_client):
    '''Paramètres :
    table : liste de dictionnaires, représentant les réservations
    numero_client : une entier, représentant le numéro du client concerné
    Valeur renvoyée : un entier donnant le nombre d'occurrences
    du numéro de client concerné'''
```

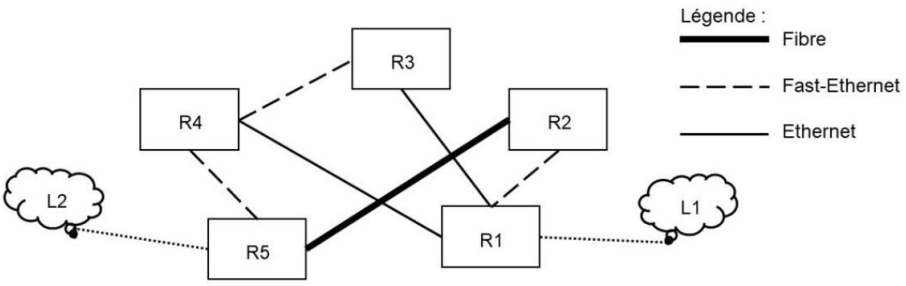
Corrigé

Voici le code complété :

```
def nombre_reservation(table,numero_client):
    '''Paramètres :
    table : liste de dictionnaires, représentant les réservations
    numero_client : une entier, représentant le numéro du client concerné
    Valeur renvoyée : un entier donnant le nombre d'occurrences
    du numéro de client concerné'''
    compteur =0
    for ligne in table:
        if ligne['num_client'] == numero_client:
            compteur+=1
    return compteur
```

4. **Protocole OSPF**

La sauvegarde de la base de données est stockée sur le réseau local L1, relié au routeur R1 du réseau suivant.

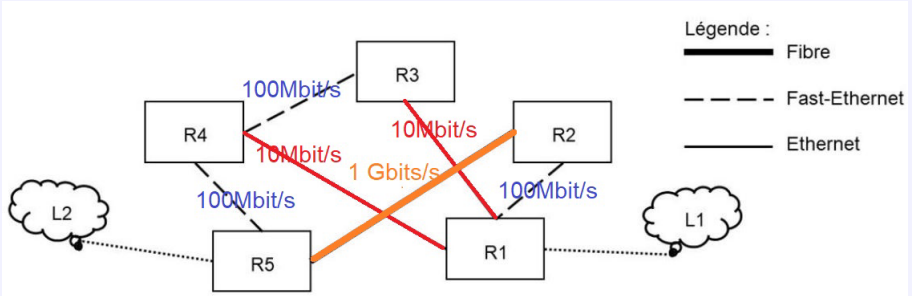


L'ordinateur de bureau du responsable de la pension fait partie du réseau local L2. Les réseaux locaux L1 et L2 font partie d'un réseau constitué de 5 routeurs (R1, R2, R3, R4, R5), de liaisons de communication dont les bandes passantes sont de 1 Gbit/s pour la Fibre, 100 Mbit/s pour Fast-Ethernet et 10 Mbit/s pour Ethernet. On s'intéresse ici au protocole de routage OSPF. Le protocole OSPF cherche à minimiser la somme des coûts des liaisons entre les routeurs empruntés par un paquet de données. Le coût C d'une liaison est donné par :

$$C = \frac{10^8}{d}, \quad \text{où } d \text{ est la bande passante en bit/s de la liaison.}$$

(a) Pour passer de L1 à L2, le chemin R1 - R2 - R5 utilisant la fibre a le coût le plus faible. Calculer ce coût.

Corrigé



Pour passer de L1 à L2, le chemin R1 - R2 - R5 utilisant la fibre a le coût le plus faible a un coût donnée par la formule :

$$C = \frac{10^8}{d}, \quad \text{où } d \text{ est la bande passante en bit/s de la liaison.}$$

On va donc calculer le coût pour chaque liaison :

- Pour R1-R2 : $d_1 = 10$ Mbit/s $= 10 \times 10^6$ bit/s $= 10^7$ bit/s et donc :

$$C_1 = \frac{10^8}{d_1} = \frac{10^8}{10^7} = 10$$

- Pour R2-R5 : $d_2 = 1$ Gbit/s $= 10^9$ bit/s et donc :

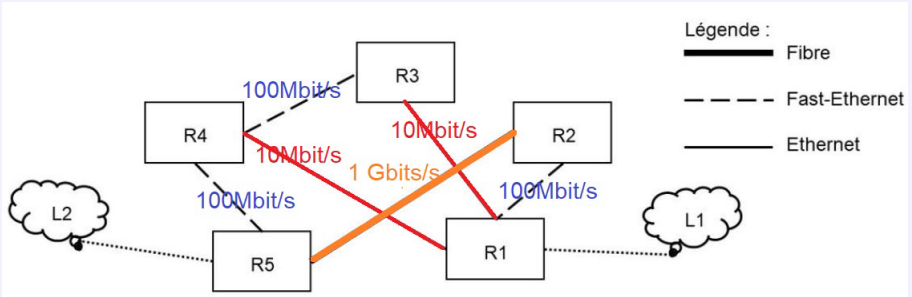
$$C_2 = \frac{10^8}{d_2} = \frac{10^8}{10^9} = 10^{-1} = 0,1$$

- Le coût total est donc :

$$\boxed{C = C_1 + C_2 = 10,1}$$

(b) La liaison entre R2 et R5 a été coupée en raison de travaux. Déterminer la route permettant de relier le réseau L1 au réseau L2 selon le protocole OSPF. Justifier.

Corrigé



De R1 on ne peut plus aller en R2 car il est isolé, on doit donc aller en R3 ou R4.

Deux chemins sont donc possibles : R1 - R3 - R4 - R5 ou R1 - R4 - R5 Le chemin au coût minimal est donc clairement le deuxième car il utilise une partie seulement du premier :

$$R_1 - R_4 - R_5$$

Exercice 3. Piles, des arbres et de l'algorithmique. 4 points

Dans cet exercice, on s'intéresse à la notation polonaise inversée (NPI) d'une expression mathématique. Dans cette notation, l'opérateur est placé après les nombres sur lesquels il s'applique. On se limitera aux expressions faisant intervenir des nombres entiers et les quatre opérateurs : $+, -, \times, /$.

Par exemple, l'expression $4 \times (5+7)$ s'écrit, en NPI : 4 5 7 + ×

L'évaluation de l'expression $12/2-4+5\times 3$, écrite en NPI : 12 2 / 4 5 3 × + est détaillée ci-dessous. Cette expression s'évalue à 17 de la manière suivante :

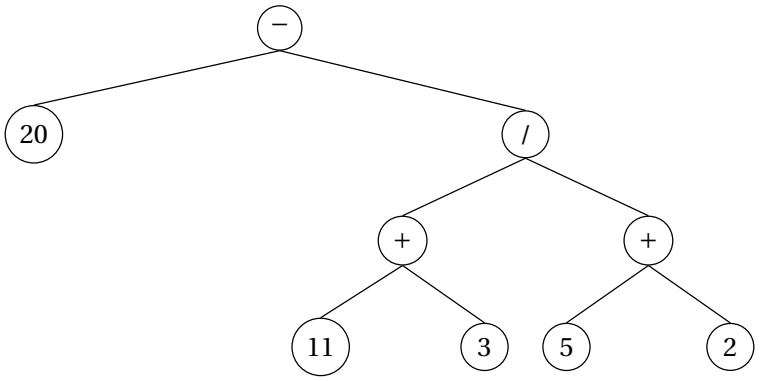
- lorsqu'on rencontre un premier opérateur $+, -, \times, /$, on évalue l'opération avec les deux nombres situés juste avant cet opérateur :
$$\underbrace{12\ 2\ /}_{12/2=6}\ 4\ -\ 5\ 3\ \times\ +$$
- on remplace cette opération par le résultat :
$$6\ 4\ -\ 5\ 3\ \times\ +$$
- on continue la lecture de l'expression :
$$\underbrace{6\ 4\ -}_{6-4=2}\ 5\ 3\ \times\ +\ \text{ devient }\ 2\ \underbrace{5\ 3\ \times}_{5\times 3=15}\ +\ \text{ devient }\ 2\ 15\ +\ \text{qui vaut 17.}$$

1. Donner la valeur de l'expression suivante écrite en NPI :
$$15\ 5\ -\ 4\ 12\ +\ \times$$

Corrigé

On part de :
$$\underbrace{15\ 5\ -}_{15-5=10}\ 4\ 12\ +\ \times$$
Donne
$$10\ 4\ 12\ +\ \times$$
Puis
$$10\ \underbrace{4\ 12\ +}_{4+12=16}\ \times$$
Soit
$$10\ 16\times$$
Puis
$$\underbrace{10\ 16\times}_{10\times 16=160}$$
D'où le résultat : 160

2. On donne l'arbre binaire suivant :



Indiquer, parmi les quatre différents parcours d'arbre ci-dessous celui qui permet d'obtenir la succession des symboles correspondant à la notation NPI suivante :

$$20\ 11\ 3\ +\ 5\ 2\ +\ /\ -$$

- un parcours en largeur ;
- un parcours en profondeur préfixe ;
- un parcours en profondeur infixe ;
- un parcours en profondeur postfixe (ou suffixe).

Corrigé

Remarque

- L'ordre préfixe : on liste chaque sommet la première fois qu'on le rencontre dans la balade.
  (préfixe = 1re rencontre ou à "gauche" dans le sens de parcours (sens anti horaire) ;)
- L'ordre postfixe (suffixe) : on liste chaque sommet la dernière fois qu'on le rencontre.
  (suffixe = dernière rencontre ou à "droite" (avec fils fantômes) dans le sens de parcours (sens anti horaire) ;)
- L'ordre infixe : on liste chaque sommet ayant un fils gauche la seconde fois qu'on le voit et chaque sommet sans fils gauche la première fois qu'on le voit.
  (infixe = 2ème rencontre ou en "dessous" (avec fils fantômes) dans le sens de parcours (sens anti horaire))

Donc ici il faut parcourir l'arbre dans un un parcours en profondeur postfixe (ou suffixe) car on liste chaque sommet la dernière fois qu'on le rencontre.

3. On utilisera une liste de symboles pour écrire la notation polonaise inversée d'une expres-sion mathématique. Par exemple, $6\ 2\ 3\ +\ \times\ 94\ 1\ -\ +$ sera représentée par la liste :

$$[6\ ,\ 2\ ,\ 3\ ,\ +\ ,\ \times\ ,\ 94\ ,\ 1\ ,\ -\ ,\ +]$$

On considère à présent l'algorithme ci-dessous, écrit en pseudo-code, qui reçoit une liste de symboles correspondant à la notation polonaise inversée d'une expression et renvoie l'arbre binaire associé.

```
Algorithme créer_arbre(liste_symboles)
P ← créer une pile vide
pour chaque élément de liste_symboles :
    si l'élément est un entier :
        a ← construire l'arbre de racine élément et n'ayant pas de sous-arbre droit ni gauche
    sinon : # l'élément est le symbole d'un opérateur
        a_droit ← dépiler P
        a_gauche ← dépiler P
        a ← construire l'arbre de racine élément,
                ayant pour sous-arbre gauche a_gauche et pour sous-arbre droit a_droit
        empiler a dans P
a ← dépiler P
renvoyer a
```

L'algorithme précédent nécessite l'utilisation d'une pile, dont les éléments sont des arbres. On précise que :

- l'action empiler a dans P insère l'élément a en haut de la pile P ;
- l'action dépiler P renvoie la valeur de l'élément en haut de la pile P et le supprime de la pile.

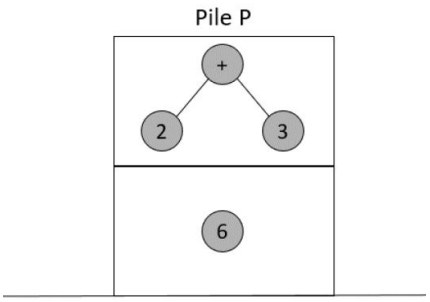
(a) Des deux acronymes LIFO et FIFO, indiquer lequel permet de décrire, de manière générale, la structure de pile. Donner la signification des quatre lettres qui composent cet acronyme.

Corrigé

L'acronyme LIFO signifie "Last In, First Out" (dernier entré, premier sorti), et décrit de manière générale la structure de pile. Cela signifie que le dernier élément ajouté à la pile sera le premier à en sortir, comme si la pile était une pile d'assiettes.

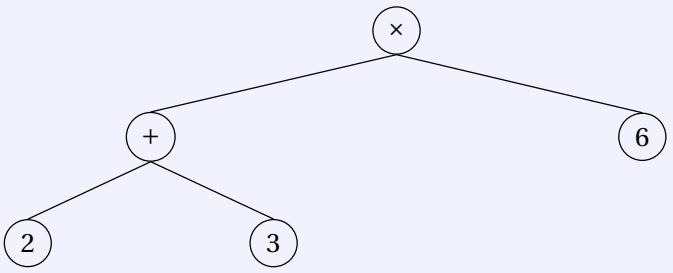
En revanche, l'acronyme FIFO signifie "First In, First Out" (premier entré, premier sorti) et décrit de manière générale la structure de file. Dans une file, le premier élément ajouté sera le premier à en sortir, comme dans une file d'attente.

(b) On applique l'algorithme précédent sur la liste [6 , 2 , 3 , + , × , 94 , 1 , − , +].
À la fin de l'exécution du quatrième tour de la boucle pour, l'état de la pile est le suivant :



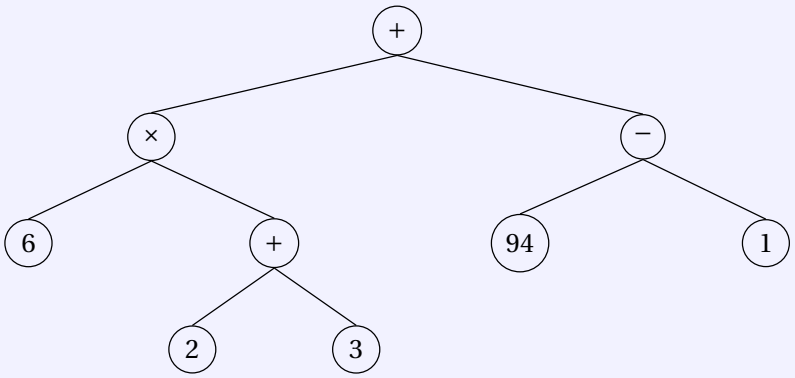
Indiquer le nombre d'élément(s) contenu(s) dans la pile P et dessiner l'état de la pile, à la fin de l'exécution du cinquième tour de la boucle pour.

Corrigé



(c) Dessiner l'arbre binaire représentant l'expression mathématique dont la notation polo-naise inversée est la liste [6 , 2 , 3 , + , × , 94 , 1 , − , +].

Corrigé



4. On souhaite écrire une fonction en langage Python qui permet d'évaluer une expression mathématique écrite avec la notation polonaise inversée NPI. On suppose qu'une telle expression est représentée par un arbre binaire obtenu à l'aide de l'algorithme précédent. On dispose pour cela de quatre fonctions :
   - est_vide(arb) qui renvoie True si l'arbre binaire arb est vide, False sinon ;
   - racine(arb) qui renvoie la valeur de la racine de l'arbre binaire arb ;
   - gauche(arb) qui renvoie le sous-arbre gauche de l'arbre binaire arb ;
   - droit(arb) qui renvoie le sous-arbre droit de l'arbre binaire arb .

   La fonction evaluer donnée ci-dessous est écrite en Python. Cette fonction prend en paramètre un arbre binaire arb représentant une expression mathématique écrite en NPI et renvoie la valeur de cette expression.

```
def evaluer (arb):
    if est_vide (gauche (arb)) and a compléter (instruction 1):
        res = a compléter (instruction 2)
    elif a compléter (instruction 2) == "+":
        res = evaluer(a compléter (instruction 3))
                + a compléter (instruction 4)
    elif a compléter (instruction 2) == "-":
        res = evaluer(a compléter (instruction 3))
                - a compléter (instruction 4)
    elif a compléter (instruction 2) == "*":
        res = evaluer(a compléter (instruction 3))
                * compléter (instruction 4)
    else:
        res = evaluer(a compléter (instruction 3))
                / a compléter (instruction 4)
    return res
```

Quatre instructions seulement permettent de compléter ce code : instruction 1, instruction 2, instruction 3 et instruction 4.
Écrire sur la copie le code de chacune de ces quatre instructions.

Corrigé

On obtient :

```
def evaluer(arb):
    if est_vide(gauche(arb)) and est_vide(droit(arb)):
        res = racine(arb)
    elif racine(arb) == "+":
        res = evaluer(gauche(arb)) + evaluer(droit(arb))
    elif racine(arb) == "-":
        res = evaluer(gauche(arb)) - evaluer(droit(arb))
    elif racine(arb) == "*":
        res = evaluer(gauche(arb)) * evaluer(droit(arb))
    else :
        res = evaluer(gauche(arb)) / evaluer(droit(arb))
    return res
```

Fin du devoir