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

Centres étrangers - juin 2021 - sujet 1 (corrigé)

Exercice 1 (POO)

1. On doit décaler chaque lettre de trois rangs vers la droite (en étant circulaire), donc la deuxième ligne affiche `'D'` et la troisième ligne affiche `'A'`.
2. La fonction suivante convient :

```
def cryptage(self,texte):
    c = ""
    for l in texte:
        c = c + self.decale(l)
    return c
```

3. Le code suivant convient :

```
cle = input("saisir la clé de chiffrement : ")
cle = int(cle)
c = CodeCesar(cle)
txt = input("saisir le texte à chiffrer : ")
print("le message chiffré est : "+c.cryptage(txt))
```

4. L'instruction va afficher `'FIN'`.

Exercice 2 (Dictionnaires)

1. (a) L'instruction renvoie : `{'type': 'classique', 'etat': 1, 'station': 'Coliseum'}`
   (b) L'instruction renvoie : `0`
   (c) L'instruction renvoie une erreur, car la clé `99` n'existe pas dans le dictionnaire `flotte`.
2. (a) Le paramètre `choix` peut être égal à `'electrique'` ou `'classique'`.
   (b)
   - ⋆ Dans le cas où le paramètre `choix` est égal à `'electrique'`, la fonction `proposition` renvoie `'Prefecture'` ou `'Jacobins'` selon la version de `Python` utilisée.
   - ⋆ Dans le cas où le paramètre `choix` est égal à `'classique'`, la fonction `proposition` renvoie `'Baraban'` ou `'Coliseum'` selon la version de `Python` utilisée.
3. (a) La fonction suivante convient :

```
def affiche():
    tab = []
    for v in flotte:
        if flotte[v]['station'] == 'Citadelle':
            tab.append(v)
    print(tab)
```

   (b) La fonction suivante convient :

```
def affiche():
    tab = []
    for v in flotte:
        if flotte[v]['type'] == 'electrique' and flotte[v]['etat'] != -1 :
            tab.append((v,flotte[v]['station']))
    print(tab)
```

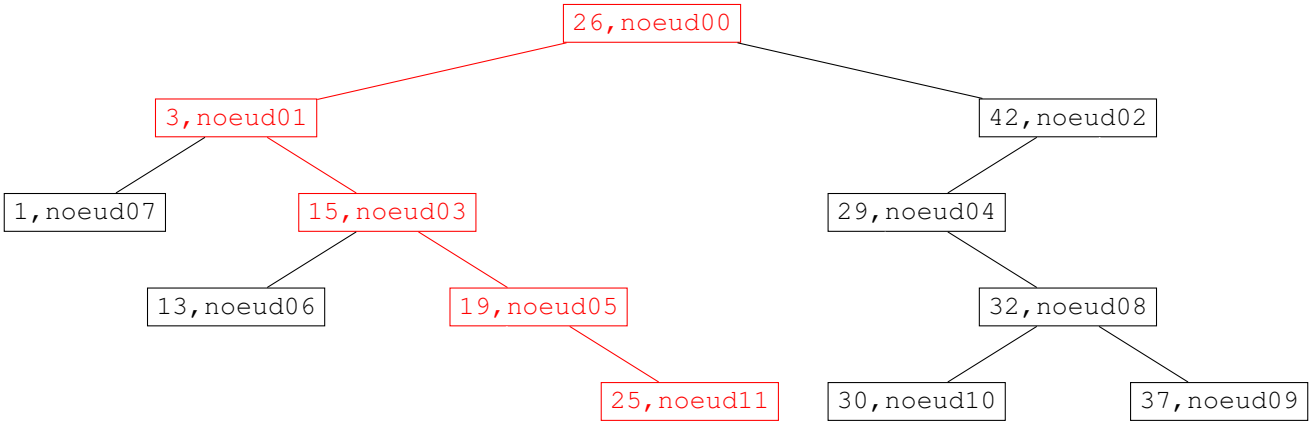
4. La fonction suivante convient :

```
def station(coord):
    d = {}
    for num,info in flotte.items() :
        nom_station = info['station']
        distance_station = distance(stations[nom_station],coord)
        if info['etat'] == 1 and distance_station < 800:
            if nom_station not in d :
                d[nom_station] = [distance_station, [num]]
            else :
                d[nom_station][1].append(num)
    return d
```

Exercice 3 (Arbres binaires de recherche)

1. On désire insérer le `noeud11` (valeur `25`) :
   - ⋆ On part de la racine (`noeud00` de valeur `26`).
   - ⋆ Comme `25` est plus petit que `26`, on considère donc le sous-arbre gauche et on se retrouve au niveau du `noeud01` (valeur `3`).
   - ⋆ Comme `25` est plus grand que `3`, on considère donc le sous-arbre droit du `noeud01` et on se retrouve au niveau du `noeud03` (valeur `15`).
   - ⋆ Comme `25` est plus grand que `15`, on considère donc le sous-arbre droit du `noeud03` et on se retrouve au niveau du `noeud05` (valeur `19`).
   - ⋆ Comme `25` est plus grand que `19`, on considère donc le sous-arbre droit du `noeud05`. Ce sous-arbre droit est vide et on insère donc le `noeud11` à cet emplacement.

   Le `noeud11` est donc inséré sous le `noeud5` en fils droit et on a l'arbre suivant :

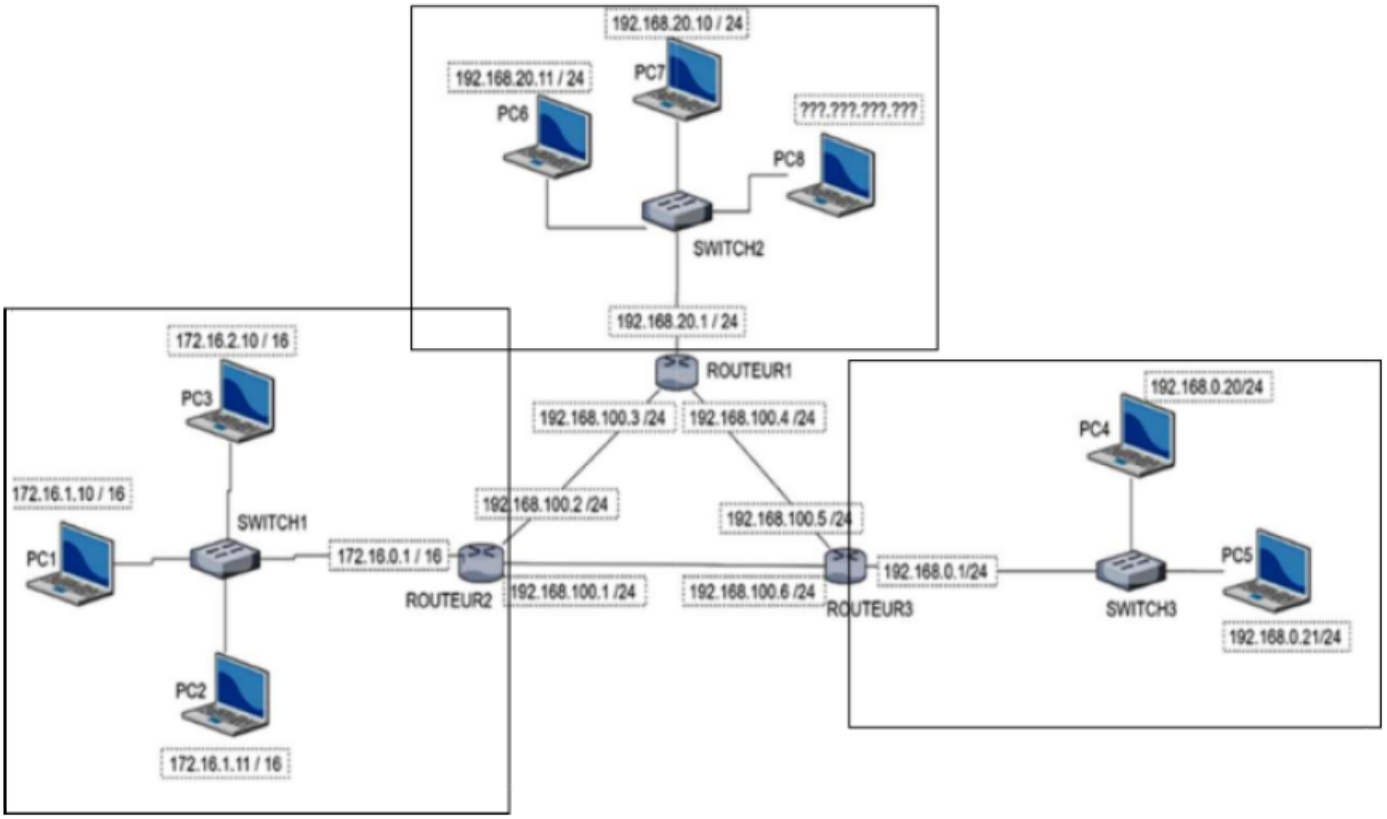


2. Comme on est dans le sous-arbre droit du `noeud00` (valeur `26`) et dans le sous-arbre gauche du `noeud04` (valeur `29`), alors il est possible de stocker toutes les valeurs comprises entre `26` et `29`, c'est à dire : `27` et `28` (les valeurs sont supposées distinctes dans l'arbre).
   (a) On a l'affichage suivant : `26 - 3 - 1 - 15 - 13 - 19 - 25 - 42 - 29 - 32 - 30 - 37`
   (b) Il s'agit d'un parcours préfixé.
3. Pour afficher les valeurs dans l'ordre croissant, il faut réaliser un parcours infixé. On a alors l'algorithme suivant :

```
fonction Parcours2(A)
    # A est un arbre binaire de recherche
    Parcours2(A.fils_gauche)
    Afficher(A.valeur)
    Parcours2(A.fils_droit)
fin fonction
```

Exercice 4 (Réseaux)
Partie A : étude de l'adressage IP

1. On a le schéma suivant :



2. (a) On utilise quatre octets dans une adresse IP V4
(b)-(e) On a le tableau suivant :

PC7	192								168								20								10							
	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0
Mas.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
	255								255								255								0							
Pour obtenir l'adresse réseau binaire, on réalise un `ET` logique entre chaque bit de l'adresse IP (ligne 2) et du masque de sous-réseau (ligne 3)																																
Rés.	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
	192								168								20								0							

3. Seules deux adresses IP sont possibles : `192.168.20.30` et `192.168.20.230`. En effet :
   - la première est l'adresse réseau, donc n'est pas autorisée ;
   - la deuxième et la cinquième ne sont pas valides car elles ont une valeur strictement supérieure à 255 ;
   - la dernière est dans le réseau `192.168.27.0` qui n'est pas le bon réseau.

Partie B : une fonction pour convertir une adresse IP en décimal pointé en notation binaire.
Les fonctions suivantes conviennent :

```
def IP_bin(adr):
    conv=[]
    for o in adr:
        conv.append(dec_bin(o))
    return conv
```

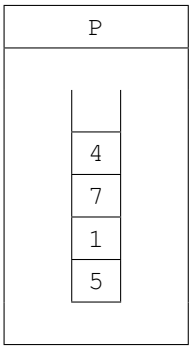
```
def IP_bin(adr):
    return [dec_bin(o) for o in adr]
```

Exercice 5 (Piles)

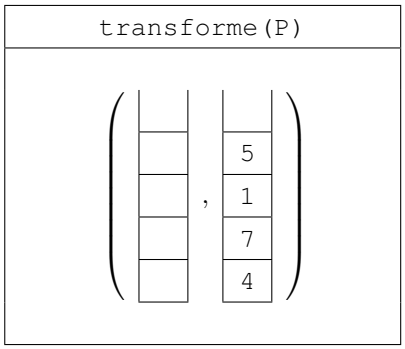
1. On a le schéma suivant :

	Etape 0 Pile d'origine `P`	Etape 1 `empiler(P,8)`	Etape 2 `depiler(P)`	Etape 3 `est_vide(P)`
				
Valeur renvoyée		`None`	`8`	`False`

2. On a le schéma suivante :



l'exécution de `transforme(P)` renvoie



3. La fonction suivante convient :

```
def maximum(P):
    m = depiler(P)
    while not est_vide(P):
        v = depiler(P)
        if v > m:
            m = v
    return m
```

4. (a) Il suffit de mettre place une boucle qui s'arrêtera quand la pile `P` sera vide. À chaque tour de boucle, on dépile `P`, on empile les valeurs précédemment dépilées dans une pile auxiliaire `Q` et on incrémente un compteur de 1. Une fois la boucle terminée, on crée une nouvelle boucle où on dépile `Q` et on empile `P` avec les valeurs dépilées (l'idée est de retrouver l'état originel de `P`). Il suffit ensuite de renvoyer la valeur du compteur.

   (b) La fonction suivante convient :

```
def taille(P):
    cpt = 0
    Q = creer_pile()
    while not est_vide(P):
        v = depiler(P)
        empiler(Q,v)
        cpt = cpt + 1
    while not est_vide(Q):
        v = depiler(Q)
        empiler(P,v)
    return cpt
```

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.