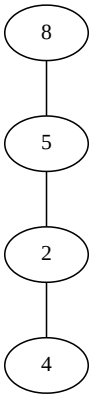
Sujet0 NSI

Exercice 1: Notion de Pile et programmation Python

Question 1: On suppose dans cette question que le contenu de la pile P est le suivant (les éléments étant empilés par le haut). Quel sera le contenu de la pile Q après exécution de la suite d'instructions suivante ?

```
Q = creer_pile_vide()
while not est_vide(P):
    empiler(Q, depiler(P))
```



> Après l'exécution de cette suite d'instruction, la pile P va se dépiler et s'empiler dans la pile Q.

Question 2:

1. On appelle hauteur d'une pile le nombre d'éléments qu'elle contient. La fonction `hauteur_pile` prend en paramètre une pile P et renvoie sa hauteur. Après appel de cette fonction, la pile P doit avoir retrouvé son état d'origine. Recopier et compléter sur votre copie le programme Python suivant implémentant la fonction `hauteur_pile` en remplaçant les `???` par les bonnes instructions.

```
def hauteur_pile(P):
    Q = creer_pile_vide()
    n = 0
    while not(est_vide(P)):
        n += 1
        x = depiler(P)
        empiler(Q, x)
    while not(est_vide(Q)):
        x = depiler(Q)
        empiler(P, x)
    return n
```

2. Créer une fonction `max_pile` ayant pour paramètres une pile P et un entier `i`. Cette fonction renvoie la position `j` de l'élément maximum parmi les `i` derniers éléments empilés de la pile P. Après appel de cette fonction, la pile P devra avoir retrouvé son état d'origine. La position du sommet de la pile est 1.

```
def max_pile(P, i):
    Q = creer_pile_vide() # création de l'instance
    sommetDeLaPile = 1
    x = depiler(P)
    elementMax = x
    empiler(Q, x)
    position = 1
    while sommetDeLaPile<1 and not(est_vide(P)):
        '''
        Tant que le sommet de la pile est inférieur à 1 et que la pile n'est pas
vide,
        ajouter 1 a la position du sommet de la pile, puis dépiler la pile P et
empiler la         pile Q avec l'élément sortant de la pile P. Et si l'élément
sortant de la pile P         (x), est supérieur à l'élément maximum, situé la
position j au même niveau que le         sommet de la pile P.
        '''
        sommetDeLaPile += 1
        x = depiler(P)
        empiler(Q, x)
        if x > elementMax:
            position = sommetDeLaPile
            elementMax = x
    while not(est_vide(Q)):
        x = depiler(Q)
        empiler(P, x)
```

Question 3:

Créer une fonction `retourner` ayant pour paramètres une pile P et un entier j. Cette fonction inverse l'ordre des j derniers éléments empilés et ne renvoie rien. On pourra utiliser deux piles auxilliaires.

```
def retourner(P, j):
    # Création des deux piles auxilliaires
    Q = creer_pile_vide()
    R = creer_pile_vide()
    nb = 0
    while nb > j and not(est_vide(P)):
        n+=1
        empiler(Q, depiler(P))
    nb = 0
    while nb < j and not(est_vide(Q)):
        n+=1
        empiler(R, depiler(Q))
    nb = 0
    while nb < j and not(est_vide(R)):
        n+=1
```

```
        empiler(P, depiler(R))
```

Question 4

L'objectif de cette question est de trier une pile de crêpes. On modélise une pile de crêpes par une pile d'entiers représentant le diamètre de chaque crêpe. On souhaite réordonner les crêpes de la plus grande (placée en bas de la pile) à la plus petite (placée en haut de la pile). On dispose uniquement d'une spatule que l'on peut insérer dans la pile de crêpes de façon à retourner l'ensemble des crêpes qui lui sont audessus. Le principe est le suivant :

- On recherche la plus grande crêpe
- On retourne la pile à partir de cette crêpe de façon à mettre cette plus grande crêpe tout en haut de la pile
- On retourne l'ensemble de la pile de façon à ce que cette plus grande crêpe se retrouve tout en bas
- La plus grande crêpe étant à sa place, on recommence le principe avec le reste de la pile.

```
def triDeCrepe(P):
    h = hauteur_pile(P)
    while h != 0:
        maximum = max_pile(P, h)
        retourner(P, maximum)
        retourner(P, h)
        h -= 1
```

Exercice 3: Arbres binaires et les arbres binaires de recherche

Question 1:

Déterminer la taille et la hauteur de l'arbre binaire suivant:

> L'arbre binaire suivant possède une taille de 9 noeuds et une hauteur de 4 noeuds.

Question 2:

On décide de numéroter en binaire les noeuds d'un arbre binaire de la façon suivante:

- La racine correspond à 1;
- la numérotation pour un fils gauche s'obtient en ajoutant le chiffre 0 à droite au numéro de son père.
- la numérotation pour un fils droit s'obtient en ajoutant le chiffre 1 à droite au numéro de son père.

1. Dans l'exemple précédent, quel est le numéro en binaire associé au nœud G?

> Le numéro en binaire associé au noeud G est le numéro 1010.

2. Quel est le noeud dont le numéro en binaire vaut 13 en décimal ?

> Le noeud dont le numéro en binaire vaut 13 en décimal est le noeud I: 1101.

3. En notant h la hauteur de l'arbre, sur combien de bits seront numérotés les noeuds les plus en bas ?

> Les noeuds les plus en bas (les feuilles de l'arbre binaire), seront numérotés sur 4 bits, puisqu'a chaque fois qu'on descend d'un niveau, on prend un bit.

Question 3:

1. Déterminer le tableau qui représente l'arbre binaire complet de l'exemple précédent.

> Le tableau représentant l'arbre binaire complet de l'exemple précédent sera représenté ainsi: [15, A, B, C, D, E, F, G, H, I, J, K , L, M, N, O].

2. On considère le père du noeud d'indice i avec *i* >= 2. Quel est son indice dans le tableau ?

> Son indice dans le tableau est *(i-1)/2* si il est impair est i/2 si il est pair.

Question 4:

On se place dans le cas particulier d'un arbre binaire de recherche complet où les nœuds contiennent des entiers et pour lequel la valeur de chaque noeud est supérieure à celles des noeuds de son fils gauche, et inférieure à celles des noeuds de son fils droit.

Écrire une fonction recherche ayant pour paramètres un arbre *arbre* et un élément *element*. Cette fonction renvoie True si element est dans l'arbre et False sinon. L'arbre sera représenté par un tableau comme dans la question précédente.

```
def recherche(arbre, element):
    tailleDeLArbre = arbre[0]
    nb = 1
    while nb < tailleDeLArbre:
        if element == arbre[nb]:
            return True
        elif element > arbre[nb]:
            nb = 1+ 2*nb
        nb = 2*nb
    return False
```

Exercice 4: Bases de données relationnelles et le langage SQL

Question 1:

1. Dans le modèle relationnel, quel est l'intérêt de l'attribut `num_eleve`

> L'attribut `num_eleve` est une clef primaire. Une clef primaire permet d'identifier une/plusieurs relations dans une table.

2. Écrire une requête SQL d'insertion permettant d'enregistrer l'élève ACHIR Mussa dans la table seconde. Les informations relatives à cet élève sont données dans la ligne 1 du fichier seconde_lyc.csv.

```
INSERT INTO seconde(num_eleve, langue1, langue2, classe)
Values (133310, 'anglais', 'espagnol', '2A');
```

3. Lors de l'insertion de l'élève ALTMEYER Yohan (ligne 2 du fichier seconde_lyc.csv), une erreur de saisie a été commise sur la première langue, qui devrait être allemand. Écrire une requête SQL de mise à jour corrigeant les données de cet élève.

```
update seconde
set langue1 = 'allemand'
where num_eleve = 156929;
```

Question 2:

1. Quel est le résultat de la requête SELECT num_eleve FROM seconde;?

> Cette requête permet d'afficher la colonne comportant essentiellement des clefs primaires (c'est-à-dire `num_eleve`) du niveau seconde.

2. On rappelle qu'en SQL, la fonction d'agrégation COUNT() permet de compter le nombre d'enregistrements dans une table. Quel est le résultat de la requête SELECT COUNT(num_eleve) FROM seconde;?

> Un enregistrement = une "ligne du tableau"
>
> Par conséquent, cette requête permet de compter le nombre de lignes pour `num_eleve`

3. Écrire la requête permettant de connaître le nombre d'élèves qui font allemand en langue1 ou langue2.

```
Select count(num_eleve) from seconde
where langue1='allemand' or langue2='allemand'
```

Question3:

1. Expliquer ce qu'apporte l'information clef étrangère pour l'attribut num_eleve de cette table en termes d'intégrité et de cohérence

> L'information clef étrangère pour l'attribut `num_eleve` permet de voir que cet attribut a été importé de la table `seconde`.

2.On suppose la table eleve correctement créée et complétée. Le chef d'établissement aimerait lister les élèves (nom, prénom, date de naissance) de la classe 2A. Écrire la commande qui permet d'établir cette liste à l'aide d'une jointure entre eleve et seconde

```sql
SELECT nom, prenom ,datenaissance
    FROM eleve
    INNER JOIN seconde
    ON seconde.num_eleve = eleve.num_eleve
    WHERE seconde.classe='2A' ;
```

Question 4:

Proposer la structure d'une table coordonnees dans laquelle on pourra indiquer, pour chaque élève, son adresse, son code postal, sa ville, son adresse mail. Préciser la clef primaire et/ou la clé étrangère en vue de la mise en relation avec les autres tables

Coordonnees
num_eleve (clef primaire, clef étrangère de la table seconde)
adresse
codepostal
ville
email

Exercice 5: Réseaux en général et les protocoles RIP et OSPF en particulier

Question 1:

1. Le routeur A doit transmettre un message au routeur G, en effectuant un nombre minimal de sauts. Déterminer le trajet parcouru

> Le routeur A peut emprunter deux trajets: ACEG ou alors ACFG. Il s'agit de la plus petite distance possible: 3 noeuds.

2. Déterminer une table de routage possible pour le routeur G obtenu à l'aide du protocole RIP

Table de routage de G		
Destination	**Routeur Suivant**	**Distance**
A	E or F	3
B	E	3
C	E or F	2
D	E	2
E	E	1
F	F	1

Question 2:

Le routeur C tombe en panne. Reconstruire la table de routage du routeur A en suivant le protocole RIP.

Table de routage de A		
Destination	**Routeur Suivant**	**Distance**
B	B	1
D	D	1
E	D	2
F	D	4
G	D	3

Question 3:

1. Vérifier que le coût de la liaison entre les routeurs A et B est 0, 01.

> La distance entre les routeurs A et B et de 10Gb/s, soit 10^{10}bits/s. Par conséquent, le coût est bien de 10^{8} / 10^{10} = 10^{-2} = 0, 01

e2. La liaison entre le routeur B et D a un coût de 5. Quel est le débit de cette liaison ?

> La distance entre le routeur B et D et de 10^{8} / 5, soit 2.10^{7} bits/s, ou encore 20 Mb/s.

Question 4:

Le routeur A doit transmettre un message au routeur G, en empruntant le chemin dont la somme des coûts sera la plus petite possible. Déterminer le chemin parcouru. On indiquera le raisonnement utilisé.

> Le chemin dont la somme des coûts est la plus petite possible est le chemin ADEG. En effet, lorsq'on convertit le coûts de chaque liaison en décimale, on obtient 0,01 + 0,001 + 1. e