Amérique du nord - mars 2023 - sujet 2 (corrigé)

Exercice 1 (Bases de données et protocoles de routage)
Partie A

1. (a) Cette requête renvoie les noms, prénoms et numéros des joueuses françaises.
   (b) La requête suivante convient :

```
SELECT nom, prenom FROM JOUEUSE WHERE pays = "Argentine" AND age >= 30
```

2. (a) La requête suivante convient :

```
UPDATE JOUEUSE SET age = 33 WHERE idjoueuse = 101
```

   (b) La requête suivante convient :

```
INSERT INTO JOUEUSE
VALUES(105, 'Angleterre', 'Warm', 'Suzanna', 29, 6)
```

3. (a) La requête suivante convient :

```
SELECT JOUEUSE.nom
FROM JOUEUSE
JOIN SELECTION ON JOUEUSE.idjoueuse = SELECTION.idjoueuse
WHERE SELECTION.points >= 10
```

   (b) La requête suivante convient :

```
SELECT COUNT(idmatch) FROM SELECTION WHERE idjoueuse = 305
```

   (c) La requête suivante convient :

```
SELECT DISTINCT MATCH.stade
FROM MATCH
JOIN SELECTION ON MATCH.idmatch = SELECTION.idmatch
WHERE SELECTION.idjoueuse = 305
```

Partie B

4. On a le trajet suivant : G - E - C - B
5. On a la table de routage suivante :

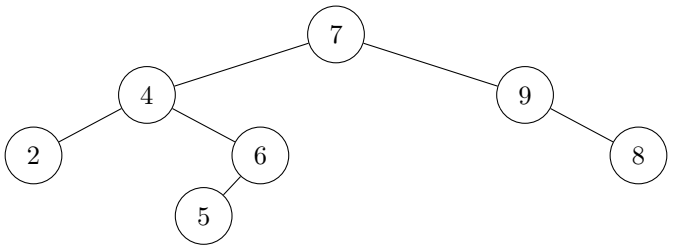
Table du routeur F de l'Amérique du Sud		
Destination	Routeur suivant	Distance
A	A	1
B	A	3
C	A	2
D	D	1
E	D	2
G	D	2

6. Il suffit que le routeur D tombe en panne pour que toutes les données entre les routeurs G (Afrique) et F (Amérique du Sud) passent par le routeur C (Asie).

Exercice 2 (Arbres binaires et ABR)

Partie A

1. L'arbre `A` a pour hauteur 4 et pour taille 12.
2. (a) On a l'arbre binaire suivant :



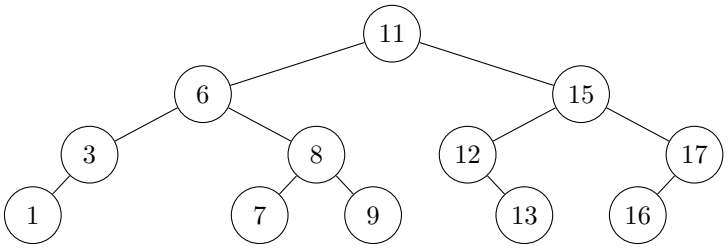
(b) Il faut saisir les instructions suivantes :

```
cons(12,
     cons(10,
          cons(8, arbre_vide(), arbre_vide()),
          cons(11, arbre_vide(), arbre_vide())),
     cons(14,
          arbre_vide(),
          cons(6, arbre_vide(), arbre_vide())))
```

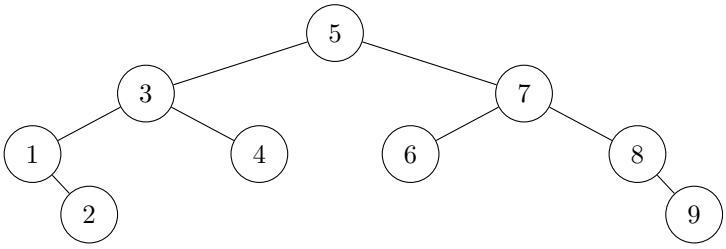
3. (a) On obtient les valeurs suivantes : $10-15-7-17-8-11-4-20-6-12-14-9$.
   (b) Lors d'un parcours infixe, on traite les nœuds dans l'ordre *gauche - racine - droite*. Il s'agit donc de la proposition 2.

Partie B

4. ⋆ L'arbre 1 n'est pas un arbre binaire de recherche car le nœud de valeur 9 est dans le sous-arbre droit du nœud de valeur 11, alors qu'il devrait être dans son sous-arbre gauche.
   ⋆ L'arbre 2 n'est pas un arbre binaire de recherche car le nœud de valeur 4 est dans le sous-arbre droit du nœud de valeur 5, alors qu'il devrait être dans son sous-arbre gauche.
5. Il faut parcourir l'arbre avec un parcours infixe puisque l'on traite les nœuds de gauche à droite.
6. (a) On obtient l'arbre suivant :



(b) On obtient l'arbre suivant :



7. La fonction suivante convient :

```
def tri(T):
    arbre = cons(T[0], arbre_vide(), arbre_vide())
    for i in range(1, len(T)):
        inserer_dans_ABR(arbre, T[i])
    return parcours(arbre)
```

Exercice 3 (Système d'exploitation, programmation générale et dictionnaires)

1. (a) `camille` étant un répertoire, il faut utiliser la commande `ls` pour afficher son contenu. Il s'agit donc de la proposition 3.
   (b) Pour accéder au répertoire `cours` d'Alix à partir du répertoire courant `camille`, il faut se déplacer successivement dans les répertoires `home`, `alix` et `cours` à l'aide de la commande `cd`. Il s'agit donc de la proposition 1 (ou 4).
   (c) Il faut utiliser l'instruction `cp fich2.txt ../camille/devoirs/fich2.txt`.
2. (a) Camille n'a accès qu'en lecture seule au fichier `DS1.txt`. Elle ne peut donc pas le supprimer en utilisant la commande `rm`.
   (b) Camille doit saisir l'instruction `chmod u+x go-r algo.py`
3. (a) La fonction suivante convient :

```
def classement_type(L):
    photos = []
    sons = []
    videos = []
    for fichier in fichiers_vacances :
        if fichier['type'] == 'photo':
            photos.append(fichier)
        elif fichier['type'] == 'son':
            sons.append(fichier)
        else:
            videos.append(fichier)
    return photos, sons, videos
```

   (b) La fonction suivante convient :

```
def photo_juillet(L):
    nb_photos = 0
    for fichier in L:
        if fichier['type'] == 'photo' and fichier['mois'] == 7:
            nb_photos += 1
    return nb_photos
```

   (c) Une démarche algorithmique possible :
   - ⋆ trier la liste `fichiers_vacances` par ordre croissant sur la taille des fichiers ;
   - ⋆ déterminer le premier indice `n` telle que la somme des tailles des éléments situés entre les indices `0` et `n` soit strictement supérieur à `64000` ;
   - ⋆ copier sur la clé USB tout les fichiers compris entre les indices `0` et `n-1`.