BAC GÉNÉRAL 2022
Épreuve de spécialité Numérique Sciences Informatiques (NSI)
Mercredi 11 mai 2022

Exercice 1

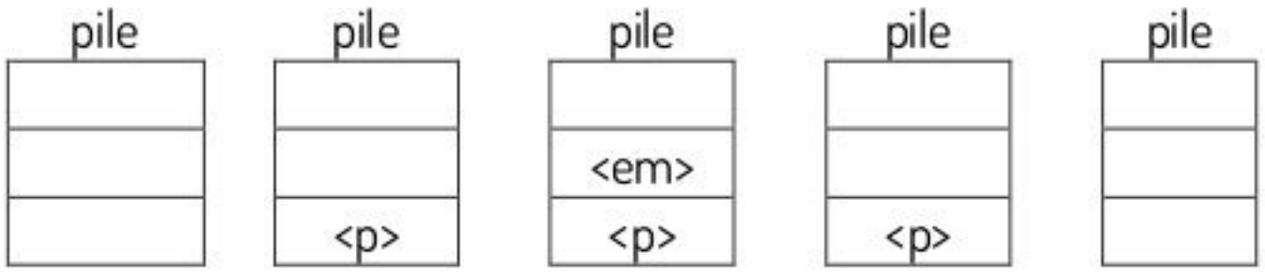
Partie A : Expression correctement parenthésée

1. La phrase décrit le comportement d'une file car la lecture se fait dans le même ordre que l'ajout, ce qui correspond à une structure de type « premier entré, premier sorti » (*First In, First Out :* FIFO).
2. Valeurs successives prises par la variable `controleur` au cours de l'analyse :
   - pour B "(((()(" : 0, 1, 2, 3, 2, 3, 2 ;
   - pour C "(()))(" : 0, 1, 2, 1, 0, -1, 0.
3. Ligne 13 :
```
if controleur < 0: # test 1
```
En effet, si la variable `controleur` passe par une valeur négative au cours de l'analyse, c'est qu'on a rencontré une parenthèse fermante de plus que de parenthèses ouvrantes déjà rencontrées. Si la condition est vérifiée, la fonction retourne False.

Ligne 16 :
```
if controleur == 0 : # test 2
```
En effet, si, à la fin de l'analyse, le compteur `controleur` est à zéro, cela signifie que l'on a rencontré autant de parenthèses fermantes que d'ouvrantes (et qu'on est arrivé à cette ligne car `controleur` n'est jamais repassé en dessous de zéro. Dans ce cas, la fonction retourne `True`.

Partie B : Texte correctement balisé

4. **a.** États successifs de la pile lors du déroulement de l'algorithme :



b. A la fin de l'analyse (hors cas d'arrêt en cours), il faut tester si la pile est vide pour pouvoir dire que le balisage est correct.

5. Si l'expression de 12 balises est correctement balisée, elle contient donc 6 paires de balises (une ouvrante et une fermante). La pile pourra donc contenir au maximum 6 éléments.

WebPédago
Partagez la connaissance !

Exercice 2

1. **a.** La requête renvoie :
   `('Crog', 'Daniel', '07-07-1968')`

   b. Requête :
   ```
   SELECT titre, id_rea
   FROM realisation
   WHERE annee > 2020
   ```
2. **a.** Il faut utiliser la requête 1 pour effectuer la mise à jour (`UPDATE`) de l'attribut `naissance` dans la table `individu`. La requête n° 2 sera refusée car elle est censée insérer un nouvel enregistrement dans la table mais avec un identifiant déjà existant. Or, cet identifiant est la clé primaire de la table et doit être nécessairement unique. La requête sera donc refusée.

   b. La relation individu peut très bien accepter deux enregistrements portant le même nom, le même prénom et la même date de naissance. La seule contrainte est que les deux identifiants (clé primaire) soient différents.
3. **a.** Demandes recopiées et complétées :
   ```
   INSERT INTO emploi
   VALUES (5400, 'Acteur(James Bond)', 688, 105);

   INSERT INTO emploi
   VALUES (5401, 'Acteur(James Bond)', 688, 325);
   ```
   b. Chaque enregistrement de la relation `emploi` doit contenir, entre autres, une référence vers la relation `realisation` (la clé du film concerné). Il est donc indispensable de créer d'abord un nouvel enregistrement pour le film 'Docteur Yes' dans la table `realisation` avant d'ajouter un enregistrement dans la table `emploi`.
4. **a.** Requête SQL affichant le nom de l'acteur, le titre et l'année de sortie à partir de tous les emplois en tant qu'acteur de James Bond :

   ```
   SELECT individu.nom, realisation.titre, realisation.annee
   FROM emploi
   JOIN individu ON emploi.id_ind = individu.id_ind
   JOIN realisation ON emploi.id_rea = realisation.id_rea
   WHERE emploi.description = 'Acteur(James Bond)';
   ```

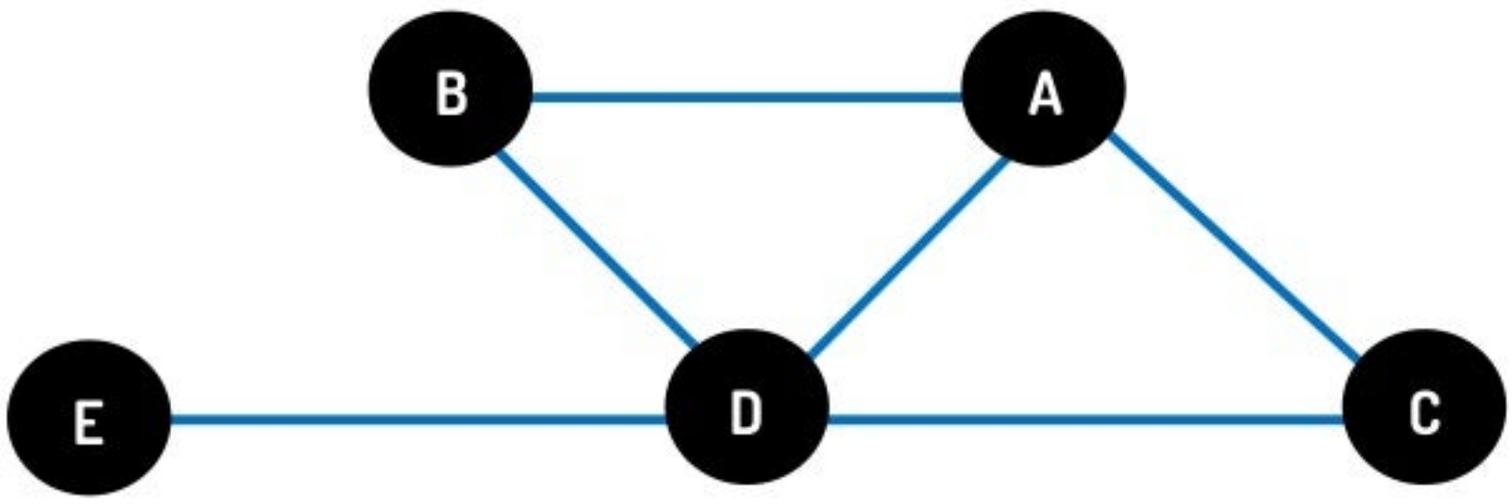
   b. Requête SQL affichant toutes les descriptions des emplois de Denis Johnson :

   ```
   SELECT emploi.description
   FROM emploi
   JOIN individu ON emploi.id_ind = individu.id_ind
   WHERE individu.nom = 'Johnson' AND individu.prenom = 'Denis'
   ```

Exercice 3

1. **a.** 11000000.10101000.10000000.10000011 s'écrit en notation décimale : 192.168.128.131.
   b. Dans le réseau A, on peut théoriquement utiliser $2^8 = 256$ valeurs différentes. Cependant, la valeur 0 est réservée à l'adresse du réseau et la valeur 255 est réservée à l'adresse de *broadcast*. Il reste finalement 254 adresses différentes possibles pour les machines du réseau A.

2. **a.** Le réseau A est relié directement aux réseaux de métrique égale à 1 dans sa table de routage, à savoir les réseaux B, C et D.
   b. Schéma des liaisons entre les cinq routeurs :



3. Tableau :

Débit	100 kbps	500 kbps	10 Mbps	100 Mbps
Métrique associée	1 000	200	10	1

4. **a.** Le chemin emprunté est celui qui a le coût le plus faible (plus court chemin dans le graphe, en utilisant l'algorithme de Djikstra). La route la plus court est donc : F – H – J – K – I (dont le coût est la somme des coûts des routes empruntées : 5 + 1 + 2 + 5 = 13).
   b. Table de routage du routeur F :

Destination	Métrique
F	0
G	8
H	5
I	13
J	6
K	8
L	11

c. Le routeur F est relié au réseau via les routeurs G (coût 10), H (coût 5, le plus petit) et I (coût 20). Si le routeur H tombe en panne, alors tous les échanges transiteront alors par le routeur G, la route G – F étant désormais celle de coût minimum.

WebPédago

Partagez la connaissance !

Exercice 4

Partie A : Parcours d'un arbre

1. Somme de l'arbre : 7 + 4 + 6 + 9 + 1 + 2 + 3 = 32.

2. Racine : A
   Feuille : C
   Nœud : B
   SAG : D
   SAD : E

3. Seule la proposition C : 3 – 6 – 2 – 7 – 4 – 9 – 1 correspond à un parcours en largeur d'abord, c'est-à-dire un parcours hiérarchique, en commençant par la racine (3), en poursuivant par tous ses enfants (6 et 2) et en terminant par tout ses petits-enfants (7, 4, 9 et 1).

4. Code de la fonction `somme` en Python :

```
def somme(tab):
    """ Entrée : tab est une liste de nombres
        Sortie : somme de tous les nombres """
    s = 0
    for i in range(len(tab)):
        s += tab[i]
    return s
```

5. La fonction `parcourir` correspond à un parcours en profondeur d'abord (en ordre préfixe, car on traite d'abord la racine et ensuite les deux sous-arbres).

Partie B : Méthode 'diviser pour régner'

6. Le principe diviser pour régner est résumé par la proposition D : diviser un problème en deux problèmes plus petits et indépendants.

7. La somme d'un arbre peut s'écrire ainsi :
   somme(arbre) = valeur(racine) + somme(SAG) + somme(SAD)

8. Code de la fonction récursive `calcul_somme` en Python :

```
def calcul_somme(arbre):
    """ Fonction récursive de calcul de somme
        d'un arbre binaire.
        Entrée : arbre
        Sortie : somme de l'arbre passé en paramètres"""
    if est_vide(arbre):
        return 0
    else:
        return valeur_racine(arbre) \
               + calcul_somme(arbre_gauche(arbre)) \
```

```
            + calcul_somme(arbre_droit(arbre))
```

Exercice 5

1. L'instruction 3 permet d'instancier correctement le joueur avec les données fournies :
   `joueur1 = Joueur("Sniper", 319, "A")`

2. **a**. Code de la méthode `redevenir_actif` en Python :

```
def redevenir_actif(self):
    """ rend le joueur à nouveau actif s'il ne l'était plus """
    if not self.est_actif:
        self.esf_actif = True
```

   b. Code de la méthode `nb_de_tirs_recus` en Python :

```
def nb_de_tirs_recus(self):
    return len(self.liste_id_tirs_recus)
```

3. **a.** C'est le test 1 qui permet de vérifier si le participant dont on souhaite récupérer les données appartient bien à l'équipe associée à la base, en comparant les valeurs (`str`) des attributs `equipe` du joueur et de la base.
   b. Le test 2 conduit à diminuer de 20 points le score de la base lorsque l'identifiant d'un tir reçu par le participant correspond à un identifiant d'un coéquipier.

4. Code à rajouter à la fin de la méthode `collecte_information` :

```
if participant.est_determine():
    self.incremente_score(40)
```