Éléments de correction du bac blanc de NSI 2025

Exercice 1

1.

Logiciels libres : Ces logiciels permettent aux utilisateurs de les utiliser, de les modifier et de les redistribuer. Ils sont souvent développés par une communauté d'utilisateurs et sont généralement gratuits. Les utilisateurs ont accès au code source, ce qui leur permet d'apporter des modifications pour répondre à leurs besoins spécifiques.

Logiciels propriétaires : Ces logiciels sont la propriété d'individus ou d'entreprises. Les utilisateurs doivent généralement payer pour les utiliser et n'ont pas accès au code source. Ils ne peuvent pas modifier le logiciel et la redistribution est généralement interdite.

En résumé, la différence clé réside dans les libertés accordées aux utilisateurs. Les logiciels libres offrent plus de liberté, tandis que les logiciels propriétaires ont tendance à avoir des restrictions plus strictes.

2. Un système d'exploitation (OS) est un logiciel qui sert d'intermédiaire entre l'utilisateur d'un ordinateur et le matériel informatique. Voici quelques-uns de ses rôles principaux :

(a) Gestion du matériel : L'OS contrôle et coordonne l'utilisation du matériel parmi les différents programmes d'application et les utilisateurs.

(b) Organisation des fichiers : Il gère les fichiers sur les dispositifs de stockage, ce qui implique leur stockage, leur récupération et leur recherche.

(c) Gestion des tâches : L'OS est responsable de l'ordonnancement des tâches et de leur exécution, ainsi que de la gestion des ressources que les tâches peuvent nécessiter.

(d) Sécurité : Il assure la sécurité des informations stockées et des ressources du système.

(e) Interface utilisateur : L'OS fournit une interface pour les utilisateurs pour interagir avec le système. Cette interface peut être une ligne de commande (CLI) ou une interface graphique (GUI).

3. Le chemin absolu du fichier `rapport.odt` dans l'arborescence de fichiers donnée est :

```
/home/elsa/documents/boulot/rapport.odt
```

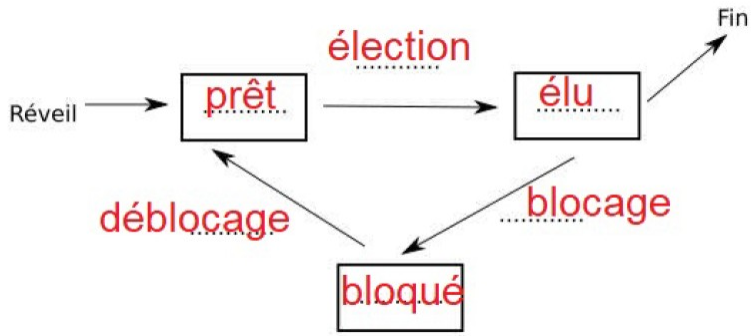
4. Le chemin relatif du fichier `photo_1.jpg` est :

```
../max/simages/photo_vac/photo_1.jpg
```

5. Après avoir exécuté la commande `cp documents/fiche.ods documents/boulot` dans la console, le fichier `fiche.ods` est copié dans le répertoire `fiche.ods` sous le même nom.

Donc le répertoire `documents` ne change pas de contenu et le répertoire `boulot` contient deux fichiers, `fiche.ods` et `rapport.odt`.

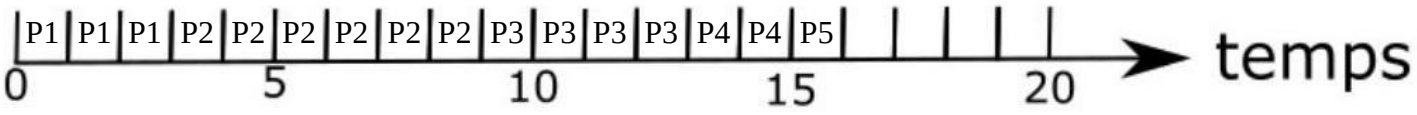
6.



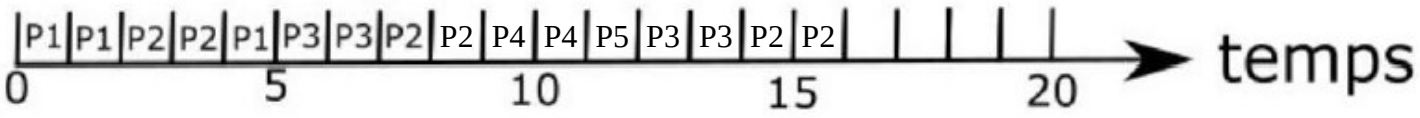
7. Supposons qu'un processus en cours d'exécution a besoin d'accéder à un fichier sur le disque dur pour lire des données. Cependant, ce fichier est actuellement utilisé par un autre processus qui l'a verrouillé pour écriture. Le processus en cours d'exécution ne peut pas continuer tant qu'il n'a pas accès au fichier. Il passe donc de l'état élu (en train d'utiliser le microprocesseur) à l'état bloqué (en attente de la libération du fichier). Une fois que le fichier est disponible, le processus peut reprendre son exécution.

8. Une structure de données linéaire de type **LIFO** (Last In, First Out) est communément appelée une **pile** . Dans une pile, les éléments sont insérés et supprimés selon le principe du dernier arrivé, premier sorti. Cela signifie que l'élément inséré en dernier est le premier à être retiré.

9.



10.



11. L'interblocage, ou deadlock en anglais, est une situation dans laquelle deux ou plusieurs processus sont incapables de progresser car chacun attend que l'autre libère une ressource. Supposons qu'il y ait deux ressources importantes, R1 et R2, et que chaque processus a besoin des deux ressources pour terminer son exécution.

Les processus P1 et P2 suivent les étapes suivantes :

(a) P1 acquiert la ressource R1.

(b) P2 acquiert la ressource R2.

Maintenant, chaque processus attend l'autre pour libérer la ressource nécessaire à son achèvement. La séquence des événements est la suivante :

- P1 détient R1 et attend R2.
- P2 détient R2 et attend R1.

Aucun des processus ne peut progresser, car chaque processus retient la ressource nécessaire pour l'autre. Cela crée une impasse, où les deux processus restent bloqués indéfiniment.

Un exemple plus concret pourrait être l'utilisation de deux imprimantes connectées à un seul ordinateur. Si P1 a déjà imprimé un document avec l'imprimante 1 et tente d'accéder à l'imprimante 2 pour imprimer un autre document, tandis que P2 a déjà utilisé l'imprimante 2 et tente d'accéder à l'imprimante 1, les deux processus peuvent se retrouver dans un état d'interblocage si les ressources ne sont pas gérées correctement.

Exercice 2

1. Voici deux services rendus par un système de gestion de bases de données relationnelles :

- intégrité des données
- optimisation des requêtes

2. bureau 1 → B → A → E → prestataire

3.

- bureau 2 → C → I → G → F → D → A → prestataire, coût = 0,1 + 1 + 0,1 + 0,1 + 1 = 2,3
- bureau 2 → C → I → H → F → D → A → prestataire, coût = 0,1 + 1 + 0,1 + 0,1 + 1 = 2,3

4. L'attribut `id_client` a été choisi comme clé primaire dans la relation `clients` car il permet d'identifier de façon unique les enregistrements de la table clients.

5.Une clé étrangère est un champ d'une table qui fait référence à la clé primaire d'une autre table afin de mettre les deux tables en relation.

6. L'erreur est que les valeurs 'Puerto sebo', 'Puerto kifecho', 'Puerto kifebo' et 'Puerto repo' n'existent pas dans la colonne de clé primaire de la table des villes.

Solution proposée : vérifier que les noms des villes dans la requête INSERT correspondent exactement aux noms des villes dans la table des villes.

7.

- La première requête SQL est utilisée pour rechercher l'identifiant (id) du client.
- La deuxième requête SQL est utilisée pour rechercher les identifiants des réservations effectuées par le client.

8.

```
SELECT reservations.id_reservation FROM clients
JOIN reservations ON clients.id = reservations.id_client
WHERE clients.nom = 'Barc'
AND clients.prenom = 'Jean'
AND clients.date_naissance = '1972-06-29'
AND clients.pays = 'Allemagne'
ORDER BY reservations.id_reservation
```

9.

```
UPDATE reservations
SET nom_croisiere = ‘Croisière Puerto’
WHERE id_reservation = 20456
```

10.

```
SELECT clients.nom, clients.prenom, clients.date_naissance
FROM clients
JOIN reservations ON clients.id = reservations.id_client
WHERE reservations.nom IN ('Croisière Piano', 'Croisière Puerto')
ORDER BY clients.nom, clients.prenom
```

Exercice 3

1.Le nœud initial est appelé **racine**
Un nœud qui n’a pas de fils est appelé **feuille**
Un arbre binaire est un arbre dans lequel chaque nœud a **au maximum** deux fils.
Un arbre binaire de recherche est un arbre binaire dans lequel tout nœud est associé à une clé qui est :

- supérieure à chaque clé de tous les nœuds de son **sous-arbre gauche**
- inférieure à chaque clé de tous les nœuds de son **sous-arbre droit**

2. Parcours **préfixe** de `arbre_no1` : **1, 0, 2, 3, 4, 5, 6**

3. Parcours **suffixe** (ou **post-fixe**) de `arbre_no2` : **0, 1, 2, 6, 5, 4, 3**

4. Parcours **infixe** de `arbre_no3` : **0, 1, 2, 3, 4, 5, 6**

5.

```
1  arbre_no1 = ABR()
2  arbre_no2 = ABR()
3  arbre_no3 = ABR()
4  for cle_a_inserer in [1, 0, 2, 3, 4, 5, 6]:
5       arbre_no1.inserer(cle_a_inserer)
6  for cle_a_inserer in [3, 2, 1, 0, 4, 5, 6]:
7       arbre_no2.inserer(cle_a_inserer)
8  for cle_a_inserer in [3, 1, 5, 0, 2, 4, 6]:
9       arbre_no3.inserer(cle_a_inserer)
```

6. Les hauteurs respectives de `arbre_no1`, `arbre_no2` et `arbre_no3` sont 5, 3 et 2.

7.

```
1 def est_present(self, cle_a_rechercher):
2     if self.est_vide() :
3         return False
4     elif cle_a_rechercher == self.cle() :
5         return True
6     elif cle_a_rechercher < self.cle() :
7         return self.sag().est_present(cle_a_rechercher)
8     else :
9         return self.sad().est_present(cle_a_rechercher)
```

8. `arbre_no3.est_presente(7)` nécessitera le moins d'appels récursif car l'ABR est équilibré → recherche en $O(ln_2(n))$

9. La différence de hauteur entre les sous arbres gauche et droit doit être inférieure ou égal à 1 pour qu'un arbre soit partiellement équilibré.

10.`Arbre_no1` n'est pas partiellement équilibré car les hauteurs de ses sous-arbres gauche et droit sont de 1 et 5, donc la valeur absolue de a différence est ágale à 4, ce qui dépasse 1.

`Arbre_no2` est partiellement équilibré car les hauteurs de ses sous-arbres gauche et droit sont toutes les deux de 3, donc la différence entre les deux est de 0, ce qui est bien inférieur à 1.

`Arbre_no3` est partiellement équilibré car les hauteurs de ses sous-arbres gauche et droit sont toutes les deux de 2, donc la différence entre les deux est de 0, ce qui est bien inférieur à 1.

11. `Arbre_no3` est le seul arbre équilibré, car il est partiellement équilibré (d'après la question précédente) et que ses sous-arbre droit et gauche sont eux aussi équilibrés (se fait par récursivité).

12.

```
1 def est_equilibre(self):
2     if self.est_vide():
3         return True
4     else:
5         equilibre = self.sad().hauteur() - self.sag().hauteur()
6         return abs(equilibre) < 2 and self.sad().est_equilibre() and
  self.sag().est_equilibre()
```