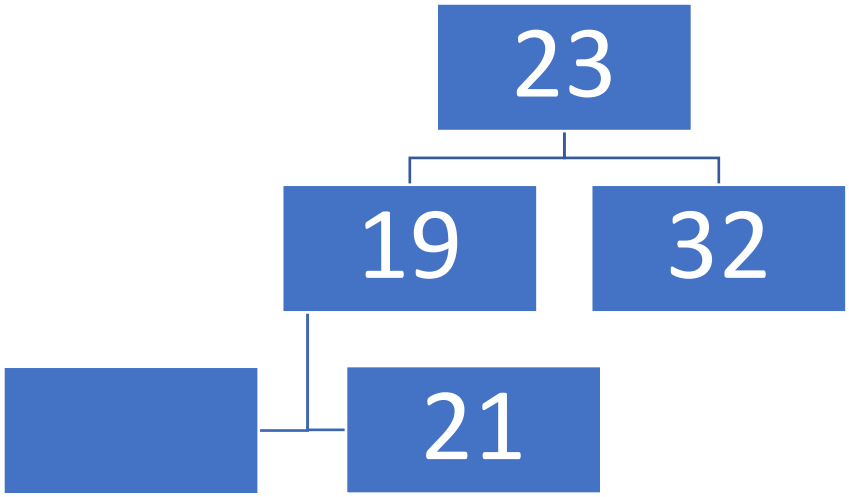
Exercice 1:

1)a)

Il y a 4 feuilles qui ont les valeurs 12 pour la première, val pour la seconde, 21 pour la troisième, 32 pour la quatrième.

1)b)

Le sous-arbre gauche du nœud 23 est:



1)c)

La hauteur de l'arbre est de 4, sa taille est de 5.

1)d)

Les valeurs entières de val pourraient être 16 ou 17 car val est plus grand que 15 et plus petit que 18.

2) a)

Le parcours infixe de l'arbre est:

15,12,13,16,18,21,19,23,32.

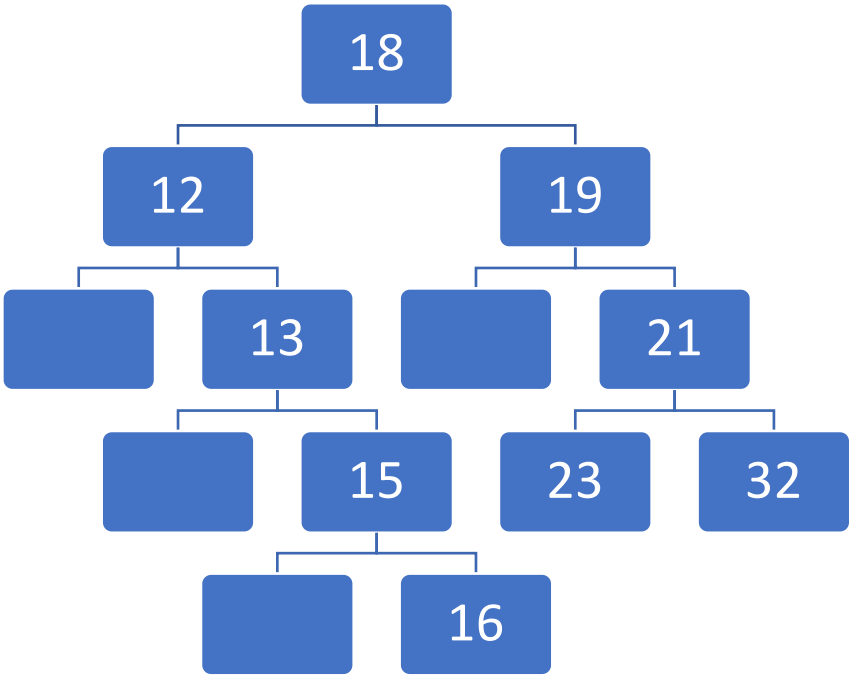
2) b)

La parcours suffixe de l'arbre est:

12,13,16,15,18,21,19,32,23

3) a)

L'arbre qui sera représenté sera:



3)b)

Les deux instructions permettant de construire l'arbre de la figure 1 sont:

```
1   if v<n.v: #si v est plus petit que le noeud parent, on insère à gauche
2       if n.ag!=None:
3           n=n.ag
4       else:
5           n.ag=Noeud(v)
6           est_insere=True
7   else: #si v est plus grand que le noeud parent, on insère à droite
8       if n.ad!=None:
9           n=n.ad
10      else:
11          n.ad=Noeud(v)
12          est_insere=True
```

3)c)

Pour insérer 19 dans l'arbre (en émettant l'hypothèse que 19 n'est pas présent dans l'arbre), le programme exécuteras le Bloc 3 puis le bloc 2 et enfin le bloc 1.

4)

```
1  def recherche(self, v):
2      tree_content=self.v
3      max_length=len(self.v)
4      tree_index=0
5      to_divide_by=2
6      while tree_index<max_length:
7          if v==tree_content[tree_index]:
8              return True
9          elif tree_index==max_length:
10             return False
11         elif v>tree_content[tree_index]:
12             tree_index+=max_length/to_divide_by
13             to_divide_by+=2
14         elif v<tree_content[tree_index]:
15             tree_index=max_length/to_divide_by
16             to_divide_by+=2
17         tree_index+=1
18     return False
```

Exercice 2:

Partie A:

1)b) ps

2)c) PID

3)b) L'ordonnancement

4)d) kill

Partie B:

1)

P3	P3	P2	P1	P1	P1	P2	P2	P3	P3
0	1	2	3	4	5	6	7	8	9	10

2)

Le Scénario 1 provoquera un interblocage.

Partie C:

1)a) Les caractères sont :

0110 0011 0100 0110

Conversion en hexadécimal:

6346

Lecture de la table:

cF

Le résultat obtenu après lecture est cF.

1)b) Chiffrage du message

Le résultat du message après chiffrage est:

$$\begin{array}{r} 0110\ 0011\ 0100\ 0110 \\ +1110\ 1110\ 1111\ 0000 \\ \hline 1000\ 1101\ 1011\ 0110 \end{array}$$

2)a) La table de vérité de l'expression booléenne (a XOR b) XOR b est:

a	b	(a XOR b)	(a XOR b) XOR b
0	0	0	0
1	0	1	1
0	1	1	0
1	1	0	1

2)b)

Comme Bob connais la chaine de caractères utilisé par Alice pour chiffrer le message, pour déchiffrer son message il doit effectuer l'opération a XOR b car c'est celle qui a été utilisée pour chiffrer le message.

Exercice 3: SQL

1)

On donne le nom générique SGBD aux logiciels qui assurent entre autres la persistance des données.

2)a)

Car sachant que l'on veut supprimer la clé primaire numT de la table Train alors qu'elle a encore des données qui lui sont reliées dans la table Reservation, il faut inverser les deux lignes pour d'abord supprimer les informations liée à la clé avant de supprimer la clé.

2) b)

Un cas pour lequel l'insertion d'information dans la table Reservation ne serais pas possible serait d'insérer une réservation pour un train inexistant.

3)a)

La commande pour donner tous les numéros de train dont la destination est "Lyon" est:

```
SELECT * FROM Train WHERE destination="Lyon";
```

3)b)

La commande pour ajouter la réservation du client Alain Turing pour le train n°654 est:

```
INSERT INTO Reservation VALUES (numR=1307, nomClient="Turing", prenomClient="Alain", prix="33€", numT=654);
```

3)c)

La commande pour mettre à jour l'horaire d'arrivée du train n°7869 est:

```
UPDATE Train;
SET VALUE (horaireArrivee="08:11") WHERE numT=7869;
```

4)

La requête permet d'afficher toutes les personnes du nom de Hopper et de prénom Grace ayant effectué une réservation.

5)

La requête permettant de renvoyer les destinations et pris des réservations effectuées par Grace Hopper est:

```
SELECT prix, destination FROM Reservation, Train WHERE nomClient="Hopper" AND prenomClient="Grace";
```

Exercice 4:

1)a)

L'ordre de grandeur du coup en nombre de comparaisons de l'algorithme de tri de fusion pour une liste de longueur n est de O^{n^2}.

1)b)

Un autre algorithme de tri est l'algorithme de tri naïf qui a un coût de O**n**2.

2)

La liste des affichages données par l'appel suivant est: [1, 2, 3, 4, 5, 6, 7, 8].

3)

Le code pour la fonction moitie_droite est:

```
1  def moitie_droite(L):
2      e=len(L)//2
3      r=len(L)
4      f=[]
5      for i in range(e,r):
6          f.append(L[i])
7      return f
```

4)

Pas réussis.

Exercice 5:

1)a)

En utilisant la table de routage, le routeur R1 enverra son message à R2 car c'est le premier dans sa liste et qu'il est en état de marche.

1)b)

Le chemin le plus court via lequel le paquet passera sera **L1-->R1-->R2-->R6-->L2** mais il y a d'autre chemins empruntable comme **L1-->R1-->R4-->R5-->R6-->L2**.

2)a)

Sachant que la liaison entre R1 et R2 est rompue, un des chemins empruntable est: **L1-->R1-->R4-->R6-->L2**, soit 5 sauts au lieu de 4.

2)b)

Dans la table de routage ci-dessus, la ligne modifiée sera celle du routeur R2 qui verras ses IPV4 disparaitre.

3)a)

La liaison R2-R3 coutera $\frac{10^9}{10*10^6}$ soit 100 bps.

3)b)

Le chemin OSPF à utiliser pour envoyer un paquet de L1 à L2 est: **R1-->R2-->R4-->R5-->R6** soit: **100+1+1+1=103**.

3)c) Les Routeurs **R2,R4,R6** verront leurs table de routage modifié selon la métrique **OSPF**.