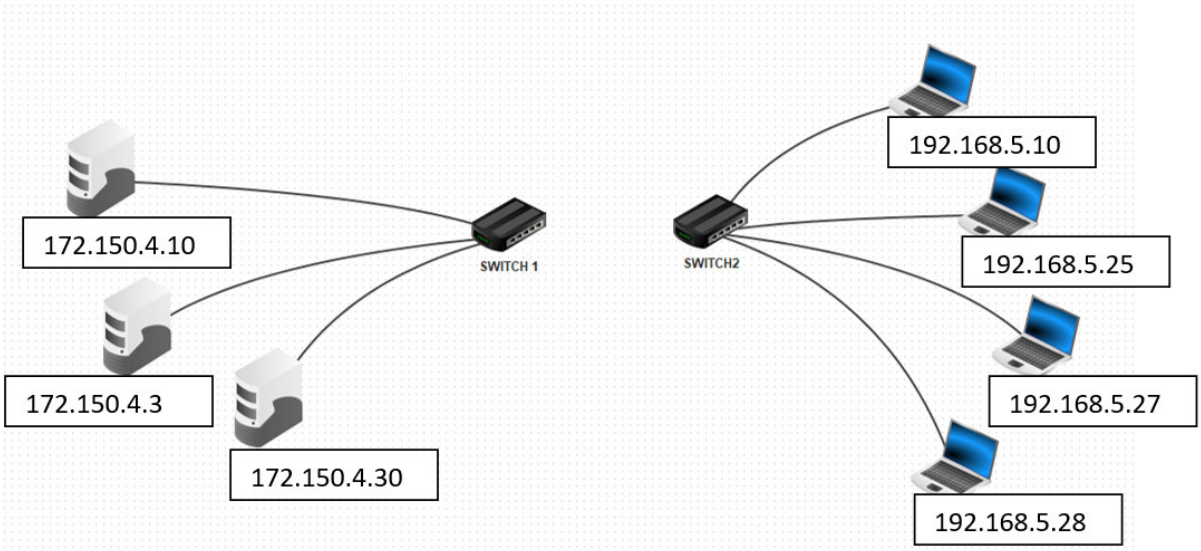
EXERCICE 5 (4 points)

Cet exercice porte sur : transmission de données dans un réseau, architecture d'un réseau, protocoles de routage, langages et programmation.

Pour une "LAN PARTY", les organisateurs gèrent deux réseaux différents non liés physiquement suivant le schéma suivant :

Réseau 1 : réseau contenant le commutateur 1 (switch1) ;

Réseau 2 : réseau contenant le commutateur 2 (switch2).



Dans cet exercice, on exploitera la notation CIDR pour l'adressage des deux réseaux.
En notation CIDR, l'adresse IP d'une machine est composée d'une adresse IPv4 et d'une indication sur le masque de sous réseau. Par exemple : 172.16.1.10 / 16 signifie :

- Adresse IP décimale : 172.16.1.10
- Masque de sous-réseau en notation CIDR : 16

La notation CIDR /16 signifie que le masque de sous-réseau a les 16 bits de poids fort de son adresse IP à la valeur 1. C'est-à-dire, pour notre exemple: 11111111.11111111.00000000.00000000.

Le PC3 du réseau 1 a pour adresse IPv4 172.150.4.30/24

1.

a. Combien d'octets sont nécessaires pour constituer une adresse IPv4 ?

b. Quelle est la notation décimale du masque de sous-réseau du PC3 du réseau 1 ?

2. Pour déterminer l'adresse IP du réseau, recopier le tableau ci-dessous et compléter les cases vides, en suivant l'ordre des instructions suivantes :

a. compléter la ligne 2 : conversion de l'adresse IP décimale en adresse IP binaire sur 4 octets ;

b. ligne 3 : compléter le masque de sous réseau en notation binaire ;

c. ligne 4 : compléter l'adresse du réseau en notation binaire suite à un ET(&) logique entre chaque bit de la ligne 2 et la ligne 3 ;

d. ligne 5 : compléter l'adresse IP décimale du réseau 1.

Adresse IP (V4) du PC3	Ligne 1	172	150	4	30
	Ligne 2	1 0 1 0 1 1 0 0	1 0 0 1 0 1 1 0		
Masque de sous réseau	Ligne 3	1 1 1 1 1 1 1 1			
Pour obtenir l'adresse réseau binaire, on réalise un ET(&) logique entre chaque bit de l'adresse IP (ligne 2) et du masque de sous réseau (ligne3)					
Adresse du réseau	Ligne 4	1 0 1 0 1 1 0 0			
	Ligne 5	172	150		

3.

a. Parmi les propositions ci-dessous, déterminer, en justifiant, celle(s) qui pourrai(en)t être utilisée(s) pour associer un 4ème PC client au réseau 1:

1) 172.154.4.30
2) 172.150.4.10
3) 172.150.10.257
4) 172.150.4.11
5) 172.150.4.0
6) 172.150.4.200

b. Quelle commande permettrait de connaître son adresse IP ?

Les organisateurs décident de faire une partie de jeu vidéo en connectant entre elles les machines des deux réseaux. Toutes les machines doivent être capables de communiquer entre elles.

4. On décide de connecter directement le switch 1 avec le switch 2 pour réaliser cette nouvelle configuration du réseau. Expliquer pourquoi cette solution n'est pas satisfaisante ? Proposer une alternative ?

5. Dans le cadre d'une future "LAN PARTY", l'organisateur veut gérer la liste des IPv4 pour éviter que deux machines aient la même adresse. Il décide de commencer son étude en créant une fonction Python `adresse`.

Une liste de listes sera utilisée pour stocker les adresses IP des machines du réseau.

Par exemple :
`liste_IP=[[192,168,10,1],[192,168,10,25],[192,168,10,13]]`

La fonction `adresse` prend en paramètres l'adresse IP (sous la forme d'une liste) que l'on souhaite tester, une liste de listes (comme `liste_IP`) et :

- si l'adresse IP testée ne figure pas dans la liste, cette fonction l'ajoute à la liste des adresses IP du réseau et affiche le message `"pas trouvée, ajoutée"` ;
- si l'adresse IP testée figure dans la liste, cette fonction se contente d'afficher le message `"trouvée"`.

Exemple :

```
>>> liste_IP=[[192,168,10,1],[192,168,10,25],[192,168,10,13]]
>>> adresse([192,168,10,3],liste_IP)
pas trouvée, ajoutée
>>> liste_IP
[[192,168,10,1],[192,168,10,25],[192,168,10,13],[192,168,10,3]]
>>> adresse([192,168,10,25],liste_IP)
trouvée
>>> liste_IP
[[192,168,10,1],[192,168,10,25],[192,168,10,13],[192,168,10,3]]
```

Ecrire en langage Python la fonction `adresse`.