TP1 : Réseaux informatique

Objectifs :

- Comprendre et manipuler les adresses IP en fonction de leur classe.
- Développer des fonctions qui vérifient la validité des adresses IP, calculent les capacités des réseaux, et déterminent l'identifiant réseau et les noms associés à des adresses IP.
- Appliquer des opérations logiques et arithmétiques simples dans un contexte de réseaux.
- Utiliser efficacement des algorithmes de recherche, en particulier la recherche dichotomique, dans la gestion des informations réseau.

Remarques générales :

- Toutes les instructions et les fonctions demandées seront écrites en C , C++, Pyton, Ou JAVA
- Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.
- Travail en groupe 2 personnes par groupe
- A rendre programmes bien détaillés avec les commentaires de chaque ligne de code
- Rapport détaillé format de chaque partie de code
- Un programme main pour tester toutes fonctionnalités.
- Un fichier Script Shell pour exécuter le programme.
- Nom de fichier sera comme suit : **Nom1 prénom1 – nom 2 prénom2 -TP1**

PROBLEME: LES ADRESSES IP DES ORDINATEURS

Présentation du problème :

Un réseau informatique est un ensemble d'ordinateurs interconnectés. Il permet la communication et l'échange des informations numériques. Il sert aussi au partage des ressources (imprimantes, disques,…) ou des applications.

Le réseau internet est un réseau mondial associant des réseaux informatiques et des ressources de télécommunication. Il est destiné à la transmission de messages électroniques, d'informations multimédias et de fichiers.

L'acheminement de ces informations sur le réseau Internet est fondé sur un protocole, nommé le protocole IP (Internet Protocol). Ce protocole permet aux ordinateurs connectés sur Internet de communiquer entre eux en utilisant des adresses numériques, appelées adresses IP.

Préliminaires :

Définition d'une adresse IP :

Une adresse IP est un numéro servant à identifier d'une façon unique chaque ordinateur connecté à Internet. Les adresses IP sont utilisées pour transmettre les données à la bonne destination.

Format des adresses IP :

Les adresses IP les plus répandues sont formées de 4 nombres. Ainsi, chaque adresse IP, est composée de 4 nombres entiers positifs ou nuls compris entre 0 et 255 (4 octets). Ces octets sont séparés par des points. Une adresse IP est notée ainsi : $octet_0.octet_1.octet_2.octet_3$ avec $0<=octet_i<=255$ pour tout i tel que $(0<=i<4)$.

Exemple d'adresse IP : 194.14.21.36 ($octet_0=194$, $octet_1=14$, $octet_2=21$, $octet_3=36$)

Notation :

Dans tout ce problème, on représentera une adresse IP par un tableau (liste sous Python) appelé **TO** de 4 nombres entiers tels que :

TO[0] représente le 1er nombre de l'adresse IP (octet0)
TO[1] représente le 2ème nombre de l'adresse IP (octet1)
TO[2] représente le 3ème nombre de l'adresse IP (octet2)
TO[3] représente le 4ème nombre de l'adresse IP (octet3)
Et l'adresse IP sera notée comme suit : TO[0].TO[1].TO[2].TO[3]

Question 1 : vérification de la validité d'une adresse IP

Soit TO un tableau de 4 entiers, on dira que TO correspond à une adresse IP valide, si pour tout i tel que (0<=i<4), on a : 0<=TO[i]<=255

✍définir une fonction d'entete **int adresseIP(int TO[4])** qui retourne 1 si TO (le tableau en paramètre) correspond à une adresse IP valide ou 0 (zéro) sinon.

Exemple :

- Si TO[0]=192, TO[1]=22, TO[2]=58 et TO[3]=168, alors l'appel de la function adresseIP(TO) retourne 1
- Si TO[0]=25, TO[1]=130, TO[2]=260 et TO[3]=87 alors l'appel de la function adresseIP(TO) retourne 0

Question 2: les classes d'une adresse IP

Une adresse IP se compose de deux parties :

- La première partie est l'identifiant réseau (en anglais NetID), elle correspond à l'adresse du réseau dans l'Internet. Elle est constitué des octets gauches de l'adresse IP (1 octet (octet 0) pour les adresses de classe A, 2 octets (octet0.octet1) pour les adresses de classe B, et 3 octets (octet0.octet1.octet2) pour les adresses de la classe C). les classes A, B et C sont définies ci-dessous
- La deuxième partie est l'identifiant machine (en anglais HostID), elle désigne l'adresse de la machine dans le réseau. Elle est constituée des octets de droite de l'adresse IP (les octets qui suivent l'identifiant réseau).

Il existe 5 classes d'adresses IP (A, B, C, D et E) selon le nombre d'octets de la partie identifiant réseau et les valeurs dans chaque octet.

On s'interessera dans ce problème, exclusivement aux trois premières classes d'adresses IP (A, B et C) qui sont utilisées dans les réseaux standards.

Une adresse IP de classe A possède les caractéristiques suivantes :

Son identifiant réseau est TO[0] avec (1<=TO[0]<=126)

Son identifiant machine est TO[1].TO[2].TO[3] avec (0<=TO[1]<=255), (0<=TO[2]<=255) et (1<=TO[3]<=254)

Ainsi une adresse IP de classe A varie de 1.0.0.1 à 126.255.255.254.

Exemples d'adresses IP de classe A :

Adresse IP	Identifiant Réseau	Identifiant machine
1.0.210.254	1	0.210.254
112.255.210.199	112	255.210.199

Question 2-a : Nombre de machines dans un réseau de classe A

On se propose de calculer le nombre maximum de machines que peut contenir un réseau dont l'adresse IP est de classe A définie plus haut)

✍ ecrire la fonction d'entête long nbMachinesA() qui retourne le nombre de tous les identifiants machines distincts 2 à 2, de classe A pour un identifiant réseau donné (TO[0] donné)

Remarque :

Pour tout identifiant réseau donné d'une adresse IP de classe A, l'identifiant machine est l'un des identifiants suivants : 0.0.1, 0.0.2,… 0.0.254, 0.1.1,…,255.255.254

Question 2-b : déterminantion de la classe d'une adresse IP

Le tableau ci-dessous résume les caractéristiques des trois classes A,B et C des adresses IP

	Identifiant réseau	TO[0]	TO[1]	TO[2]	TO[3]	Valeurs possibles
A	TO[0]	1 à 126	0 à 255	0 à 255	1 à 254	1.0.0.1 à 126.255.255.254
B	TO[0].TO[1]	128 à 191	0 à 255	0 à 255	1 à 254	128.0.0.1 à 191.255.255.254
C	TO[0].TO[1].TO[2]	192 à 223	0 à 255	0 à 255	1 à 254	192.0.0.1 à 223.255.255.254

✍définir une fonction d'entête char classe(int TO[4]) qui retourne le caractère 'A' ou 'B' ou 'C' correspondant à la classe de l'adresse IP du tableau TO donné en paramètre. si le paramètre TO ne correspond pas à une adresse IP valide (voir Question I-1) ou si TO ne désigne pas une adresse IP de classe 'A' ou 'B' ou 'C', la fonction classe(TO) retournera le caractère 'X'.

Exemple :

- Si TO[0]=194, TO[1]=252, TO[2]=18 et TO[3]=40, alors l'appel de la fonction classe(TO) retourne 'C'
- Si TO[0]=127, TO[1]=0, TO[2]=0 et TO[3]=1, alors l'appel de la fonction classe(TO) retourne 'X'

Question 3 : le masque d'un réseau (NetMask)
Un masque réseau (en anglais NetMask) se présente sous la forme de 4 octets séparés par des points (comme une adresse IP), son rôle est de reconnaitre l'identifiant réseau associé à une adresse IP en utilisant l'opération binaire « et binaire » (définie plus bas)
Question 3-a : décomposition binaire
✍définir une fonction d'entête void binaire(int octet, int bin[8]) qui met la représentation binaire du nombre octet (premièr paramètre supposé entre 0 et 255) dans le tableau bin (deuxième paramètre). chaque élement bin[i] pour i tel que 0<=i<8, contiendra 0 ou 1 selon la décomposition binaire du nombre octet qui sera ainsi : bin[7]bin[6]bin[5]bin[4]bin[3]bin[2]bin[1]bin[0] (voir exemple)
Remarque : bin[0] contiendra le bit de poids faible.
Exemple :
Soit bin un tableau de 8 entiers déjà déclaré, alors l'appel de la fonction binaire(75,bin), va mettre les bits suivants dans le tableau bin :
bin[0]=1, bin[1]=1bin[2]=0, bin[3]=1, bin[4]=0, bin[5]=0, bin[6]=1 et bin[7]=0
la décomposition binaire de 75 est notée ainsi 01001011.
Question 3-b : l'opération « etBinaire » entre 2 entiers
L'opération « etBinaire » est effectuée bit à bit sur les représentations binaires de nombres entiers selon les règles suivantes :
0"etBinaire"0=0, 0"etBinaire"1=0, 1"etBinaire"0=0, 1"etBinaire"1=1
✍soient x et y deux entiers tels que (0<=x<=255) et (0<=y<=255), écrire une fonction d'entête int etBinaire(int x, int y) qui retourne le résultat décimal (en base 10) de l'opération x "etBinaire"y (définie plus haut)
Exemple :
Si x=45 et y=138 alors l'appel etBinaire(x,y) retourne 8, x en bianire 00101101, y en binaire 10001010, x"etBinaire"y est 00001000 (8).
Question 3-c : Détermination de l'identifiant réseau
Le masque réseau comprend dans sa notation binaire des 1 aux niveaux des bits de la partie identifiant réseau de l'adresse IP et des zéros aux niveaux des bits de l'identifiant machine. Le tableau suivant donne les valeurs du masque réseau en fonction des classes d'adresse IP

Classe	Masque réseau en binaire	Masque réseau décimal
A	11111111.00000000.00000000.00000000	255.0.0.0
B	11111111.11111111.00000000.00000000	255.255.0.0
C	11111111.11111111.11111111.00000000	255.255.255.0

Pour obtenir les octets de l'identifiant réseau (NetId), il suffit d'effectuer l'opération "etBinaire" (définie dans la question I-3-b) entre les octets de l'adresse IP (TO) et les octets du masque de réseau.
✍ en utilisant l'opération "etBinaire", définir la fonction d'entête void idReseau(int TO[4], int NetId[4]) qui met dans le tableau NetId (deuxième paramètre) les octets de l'identifiant réseau de l'adresse IP représentée par TO (premier paramètre) et supposée etre de classe 'A' ou 'B' ou 'C'
Remarque : les octets non significatifs de l'identifiant réseau NetId seront des 0.
Exemple :
Soit l'adresse IP représentée par : 36.142.123.12, (TO[0]=36, TO[1]=142, TO[2]=123 et TO[3]=12), alors c'est une adresse de classe 'A' et son masque de réseau est 255.0.0.0 (voir tableau des masques réseaux définis ci-dessus).Ainsi NetId[0]=36"etBinaire"255, NetId[1]=142"etBinaire"0, NetId[2]=123"etBinaire"0, et NetId[3]=12"etBinaire"0.
Dans ce cas l'appel de la fonction idReseau(TO,NetId), va mettre les valeurs suivantes dans NetId ; NetId[0]=36, NetId[1]=0, NetId[2]=0 et NetId[3]=0

Question 4 : Le DNS (Domain Name System)
Au lieu d'utiliser directement des adresses IP numériques pour identifier des ordinateurs il est possible de les désigner par des noms de domaines, plus faciles à retenir, grâce à un système appelé DNS (Domain Name System). Le DNS permet ainsi d'associer des noms aux adresses IP numériques et d'organiser ces noms de façon plus signifiante pour l'utilisateur. Il s'agit dans ce qui suit de rechercher un nom de domaine associé à une adresse IP en utilisant la recherche dichotomique.

Question 4-a : Recherche dichotomique dans un tableau trié
Soit T un tableau de N (N>0) entiers classés par ordre croissant (pour tout i tel que (0<=i<(N-1)), on a T[i]<=T[i+1])
On recherche si x, un nombre entier quelconque, est un élément du tableau trié T en utilisant la recherche dichotomique dont le principe est le suivant :

- On calcule m le milieu des indices de l'intervalle de recherche (Au début l'intervalle de recherche est tout le tableau T et les indices varient entre 0 et (N-1))
- Si x=T[m] alors x est un élément du tableau T et on termine le traitement.
- Sinon (si x !=T[m]) on réduit l'intervalle de recherche en sélectionnant la partie de l'intervalle de recherche pouvant contenir x.

Ce traitement (calcul de m et comparaison de x avec T[m]) se répète pour chaque intervalle de recherche. Et on arrete si x est trouvé dans un intervalle de recherche, ou si x ne pourra jamais se trouver dans l'intervalle de recherche.
Exemple de déroulement d'une recherche dichotomique : soit T un tableau de 10 entiers classés par ordre croissant, comme suit : T[0]=2, T[1]=5, T[2]=8, T[3]=9, T[4]=13, T[5]=20, T[6]=32, T[7]=58, T[8]=62, T[9]=80 et on recherche x=20.

	Intervalle de recherche	m (indice du milieu)	Comparaison
1ère itération	T[0] à T[9]	m=(0+9)/2=4 (division entière)	x>T[4]
2ème itération	T[5]à T[9]	M=(5+9)/2=7	X<T[7]
3ème itération	T[5] à T[6]	M=(5+6)/2=5	X=T[5] (trouvé)

Soit T un tableau de N entiers distincts 2 à 2 et triés par ordre croissant. Soient inf et sup 2 entiers tels que 0<=inf<=sup<N et soit x un entier quelconque.
✍ définir une fonction récursive d'entête int rechercheDicho(int N, int T[], inti nf, int sup, int x). cette fonction utilise le principe de la recherche dichotomique pour rechercher x et elle retourne l'indice i (inf<=i<=sup), tel que x=T[i] ou elle retourne -1 s'il n'existe aucun i (inf<=i<=sup), tel que x=T[i].
Exemple :
Soit N=8 et T un tableau des 8 entiers suivants : T[0]=1, T[1]=4, T[2]=6, T[3]=9, T[4]=10, T[5]=15, T[6]=18, T[7]=30, si inf=2, sup=6 et x=15, alors l'appel
rechercheDicho(N,T,inf,sup,x) retournera 5.

Question 4-b : recherche du nom associé à une adresse IP
On considère la structure struct machine définie ci-dessous qui associe un nom à un identifiant machine d'une adresse IP de classe C (l'identifiant machine d'une adresse IP de classe C est composé du seul octet TO[3])

```
struct machine
      {int hostId ;  //l'identifiant machine d'une adresse IP de classe C
      Char nom[100] ; // le nom associé à l'adresse IP
      } ;
```

- Soit N une constante entière strictement positive déjà définie
- Soit tabMachines déclaré ainsi : struct machine tabMachines[N] ;

Le tableau tabMachines contient des identifiants machines des adresses IP de classe C ayant un même identifiant réseau ainsi que les noms qui lui sont associés.
On suppose que le tableau tabMachines est trié par ordre croissant des identifiants machines (hostId) et que ces identifiants machines sont distincts 2 à 2 ainsi : pour tout i tel que 0<=i<(N-1)) , on a tabMachines[i].hostId<tabMachines[i+1].hostId
✍ En utilisant la question précédente (question I-4-a), écrire la fonction d'entête : int nomMachine(int N, struct machine tabMachines[], int x, char nomR[]) qui recherche x parmi les identifiants machines du tableau tabMachines.
Ainsi si x correspond au champ hostId d'un élément du tableau tabMachine, (s'il existe i (0<=i<N) tel que x=tabMachines[i].hostId), il faut mettre dans la chaine nomR (nom recherché) le nom associé à cette adresse IP dans la structure struct machine et dans ce cas la fonction retournera l'indice i. sinon, si x ne correspond à

aucun champ hostId d'un élément du tableau tabMachine, nomR contiendra une chaine vide(« «), et la fonction retournera -1.

Exemple :

Soit tabMachines, un tableau de 4 éléments de type struct machine triés par ordre croissant du champ hostId comme suit :

Champ	tabMachines[0]	tabMachines[1]	tabMachines[2]	tabMachines[3]
hostId	65	98	205	233
nom	"machineX"	"machineA"	"machineY"	"machineB"

Soit x=205 et nomR une chaine de caractères déjà déclarée, alors l'appel de la fonction nomMachine(4,tabMachines,x,nomR) va retourner 2 (car x=tabMachines[2].hostId) et on aura nomR="machineY"