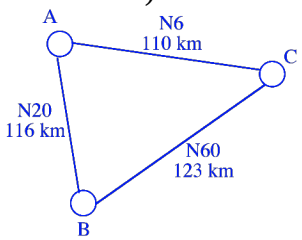
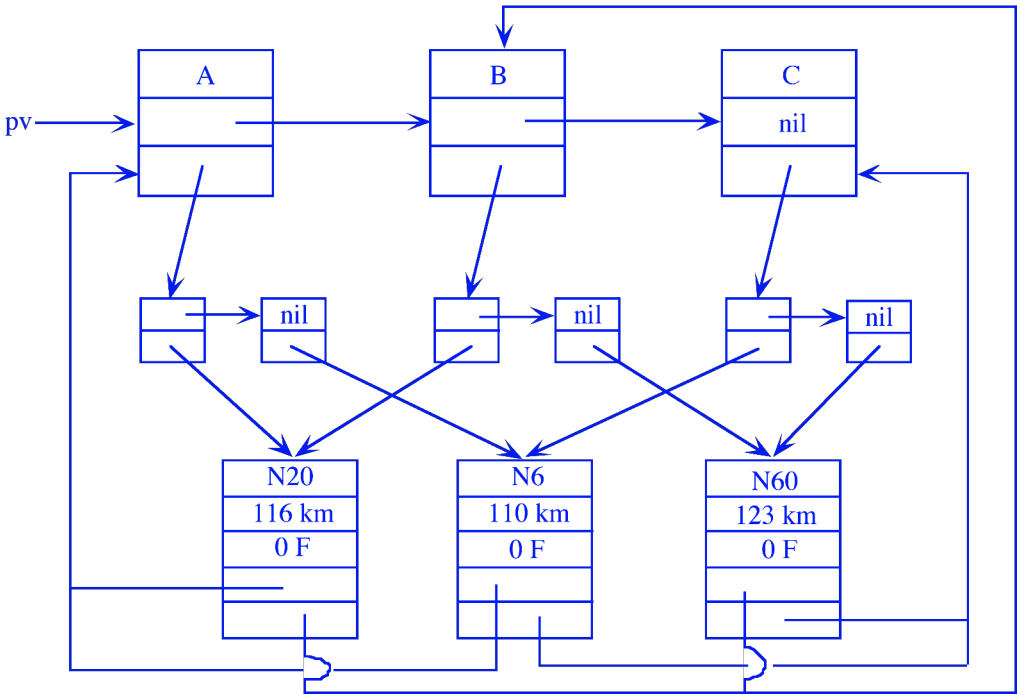
TD10 – Soutien – Villes

On représente le réseau routier d'un pays de la manière suivante : les villes sont des enregistrements chaînés les uns aux autres qui « pointent » également vers des enregistrements appelés « connexions routières » ; ces derniers sont chaînés entre eux lorsqu'ils concernent la même ville ; chaque connexion « pointe » de plus vers un « tronçon routier » qui indique le numéro de la route, la distance, le péage ainsi que l'origine et l'extrémité du tronçon (pointeurs sur des villes).



Exemple : la structure de données représentant les connexions entre les villes A, B et C donnent :



La description des structures de données utilisées est donc la suivante :

```
Type  listeVille = ^ville
      listeConRou = ^conRou
      listeTron = ^tron
      ville =     Enregistrement
                        nom : Chaîne
                        suiv : listeVille
                        connex : listeConRou
                  FinEnregistrement
      conRou =    Enregistrement
                        suiv : listeConRou
                        tron : listeTron
                  FinEnregistrement
      tron =      Enregistrement
```

```
                    route : Chaîne
                    dist, peage : entier
                    orig, extrem : listeVille
                FinEnregistrement
```

4.1. On souhaite écrire la fonction `sontReliees` qui retourne un pointeur sur un tronçon routier entre les villes a et b (données par leur nom), si un tel tronçon existe. Dans le cas contraire, la fonction retourne « nil ».
Expliquez le principe en français puis écrivez en pseudo-langage cette fonction.

```
Fonction sontReliees (pv : listeVille ; a, b : chaine) : listeTron
Var q : listeVille
    t : listeTron
    cr : listeConRou
Debut
q ← pv
t ← nil
TantQue (q≠nil) et (q^.nom≠a) Faire
    q ← q^.suiv
FinTantQue
Si q=nil
    Alors écrire('La ville', a, 'n''existe pas')
    Sinon cr ← q^.connex
          TantQue cr≠nil et (cr^.tron^.orig^.nom ≠ b ou
                                    cr^.tron^.extrem^.nom ≠ b) Faire
              cr ← cr^.suiv
          FinTantQue
          Si cr<>nil
              Alors t ← cr^.tron
          FinSi
FinSi
Retourner(t)
Fin
```

4.2. On souhaite écrire une procédure `donneVille` qui retourne le nom des deux villes reliées par un tronçon routier donné par son nom (le tronçon existe).
Expliquez le principe en français puis écrivez en pseudo-langage cette procédure.

```
Procedure donneVille (E pv : liteVille , nomtron : chaine, S a, b : chaine)
Var q : listeVille
    t : listeTron
    cr : listeConRou
    trouve : booléen
Debut
q ← pv
trouve ← faux
TantQue (q≠nil) et  non trouve Faire
    cr ← q^.connex
    TantQue (cr≠nil) et  non trouve Faire
        t ← cr^.tron
        Si t^.route = nomtron
            Alors trouve ← vrai
                  a ← t^.orig^.nom
                  b ← t^.extrem^.nom
            Sinon cr ← cr^.suiv
        Finsi
      FinTantQue
 q ← q^.suiv
FinTantQue
Fin
```