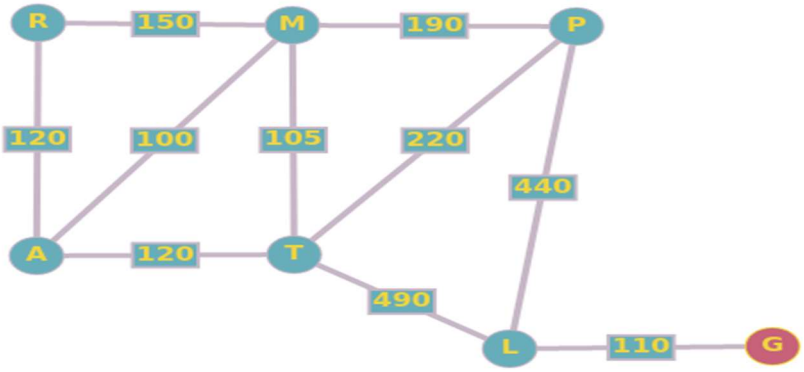
Tp réseau routier

On représente le réseau autoroutier entre les villes de Rennes (R), Angers (A), Tours (T), Le Mans (M), Paris (P), Lyon (L) et Grenoble (G) à l'aide d'un graphe.

Les villes sont les sommets du graphe et les (auto)routes sont représentées par les arêtes du graphe.

On a indiqué les distances entre les villes sur les arêtes



1. Matrice d'adjacence

Ecrire la matrice d'adjacence de ce graphe (la matrice sera représentée par un tableau à 2 dimensions, la variable s'appellera `matrice1`, les villes sont classées **dans l'ordre alphabétique)**

```
matrice1 = [[….]]

# Vérification de la réponse
assert matrice1[0][6] == 120
assert matrice1[1][2] == 110
```

2. Liste d'adjacence

Ecrire une fonction `matriceversliste(matrice, noms)`

- prenant en paramètres `matrice` : matrice d'adjacence et `noms` : noms des sommets dans l'ordre de la matrice
- renvoyant un dictionnaire dont les clés sont les sommets et les valeurs sont un tableau de tuples au format `('Nom', distance)`.

Exemple : A est reliée à M, R et T. Le dictionnaire commencera par `{'A':[('M',100), ('R', 120), ('T', 120)] …}`

```
def matriceversliste(matrice, noms):
    """Renvoie la liste d'adjacence sous
format dictionnaire"""

matrice = [[1, 2, 1, 0],
           [2, 3, 0, 1],
           [1, 0, 0, 0],
           [0, 1, 0, 4]]
test = matrice2liste(matrice, ['A', 'B', 'C',
'D'])
assert test['C'] == [('A', 1)]
assert ('B', 2) in test['A']
assert ('D', 4) in test['D']
```

3. Liste d'adjacence vers matrice

On considère à présent le graphe représentant les mêmes villes mais les sommets sont pondérés par le temps de parcours en minutes.

Voici sa liste d'adjacence au format dictionnaire tel que décrit plus haut:

```
{'A': [('M', 65),('R', 90),('T', 80)],
 'G': [('L', 70)],
 'L': [('G', 70), ('P', 230),('T', 260)],
 'M': [('A', 65),('P',95),('R',90),('T', 55)],
 'P': [('L', 230),('M', 95), ('T', 130)],
 'R': [('A', 90),('M', 90)],
 'T': [('A', 80),('L',260),('M',55),('P',130)]}
```

Ecrire une fonction `listeversmatrice(dico)` prenant en paramètre un graphe donné par une liste d'adjacence sous format dictionnaire comme ci-dessus et renvoyant la matrice d'adjacence de ce graphe ainsi que un tableau des sommets. En d'autres termes : la fonction `liste2matrice` est l'inverse de la fonction `matrice2liste` précédente.

Aide : la méthode T.index(element) récupère l'indice de élément dans T.
(si tab= [2,7,3,5] alors tab.index(3) retourne 2)

```
def listeversmatrice(dico):
    """Converti une liste d'adjacence en matrice
d'adjacence"""

# Vérification
test = {'A':[('A', 1),('B', 2),('C', 1)],
        'B': [('A', 2),('B', 3),('D', 1)],
        'C': [('A', 1)],
        'D': [('B', 1),('D', 4)] }
l,n = liste2matrice(test)
assert n == ['A', 'B', 'C', 'D']
assert l == [[1,2,1,0],[2,3,0,1],[1,0,0,0],[0,
1,0,4]]
```

Exercice : Écrire une fonction kilométrage donnant le nombre de kilomètre d'un trajet dont les villes sont données dans un tableau ou 0 si le trajet est impossible
*On utilisera la **structure** de votre choix.*
Exemple pour ['A','T','P'] la distance est 340