Exercice 1

On représente un graphe sous la forme d'une liste d'adjacence stockée dans un dictionnaire.

```
G = {
    'A': ['B', 'C'],
    'B': ['C', 'D'],
    'C': ['D'],
    'D': ['C'],
    'E': ['F'],
    'F': ['C'],
}
```

1. Représenter le graphe associé à G.
2. On considère l'algorithme récursif suivant :

```
def parcours(graph, node, visited=None):
    if visited is None:
        visited = []
    if node not in visited:
        visited.append(node)
    unvisited = [n for n in graph[node] if n not in visited]
    for node in unvisited:
        parcours(graph, node, visited)
    return visited
```

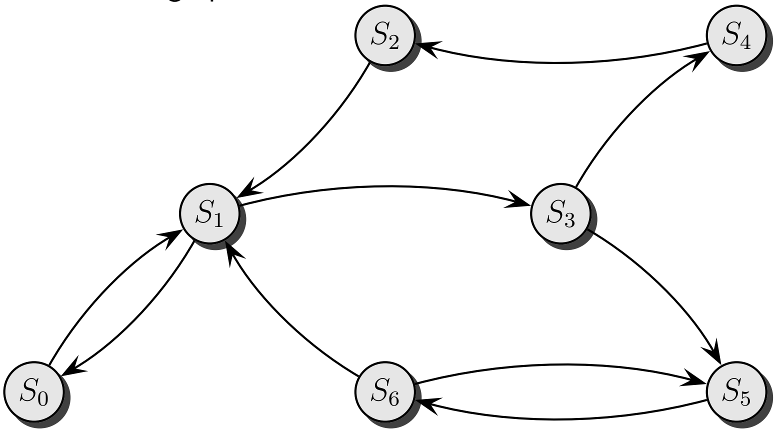
   (a) Expliquer pourquoi l'algorithme termine

   (b) S'agit-il d'un algorithme de parcours en largeur ou en profondeur ?

   (c) Appliquez-le sur l'exemple ci-dessus.

Exercice 2

On considère le graphe suivant.



1. Donner la matrice d'adjacence M du graphe précédent. On rappelle que $M_{ij} = 1$ si l'on peut aller de i vers j, 0 sinon, pour tous $i, j \in \{0, ..., 6\}$.
2. Pour $i, j \in \{0, \ldots, 6\}$, que représente le coefficient $[M^2]_{i,j}$?
3. On stocke en Python un graphe orienté sous la forme d'un dictionnaire, constitué de clés représentants des sommets et dont les valeurs sont les listes de sommet accessibles depuis la clé.
   $G = [sommet1 : [successeur11, ...successeur1n1], sommetm : [successeurm1, ...successeurmnm]]$
   Expliciter le dictionnaire correspondant au graphe ci-dessus.
4. Ecrire une fonction Python, `adjacence(G)` qui à un dictionnaire représentant un graphe G associe sa matrice d'adjacence.