X1ii020 Langages de Programmation de Haut Niveau Groupe : ..... Note : /20

N° étudiant : ..... Nom + Prénom .....

Durée 1/2h. Documents autorisés. Le barème et la durée des exercices ne sont donnés qu'à titre indicatif.

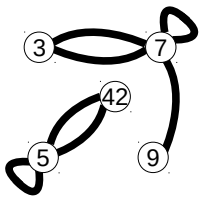
- Contrôle Continu du 18 octobre 2018

1 - Graphe Non Orienté [3 points]

Un graphe (non orienté) peut être représenté en Python par un couple (V,E) où V est l'ensemble des sommets et E la liste des arêtes (c'est une liste, pas un ensemble, car une arête peut être multiple). Chaque arête est un ensemble de sommets de cardinal 1 ou 2 : de cardinal 1 s'il s'agit d'une « boucle », de cardinal 2 sinon.

```
Gex = ( {3,7,9,5,42}, [ {5,42}, {7,9}, {7}, {7,3}, {42,5}, {5}, {3,7} ] )

def voisins(s,G) :
  if not(s in G[0]) :
    return set()  # l'ensemble vide (!!! pas {} !!!)
  else :
    return { t for t in G[0] if {t,s} in G[1] }
```




```
>>> print(voisins(7,Gex))
        {9, 3, 7}
```

```
>>> print( [b for a in Gex[0] for b in range(a) if a+b in Gex[0] ] )
        [0, 2, 0, 2, 4, 0, 2, 0, 0]
```

Compléter (colonne de droite) le tableau ci-dessous avec l'affichage produit par le code fourni.

```
print(voisins(6,Gex))

print( { a for a in Gex[0] if {a} in Gex[1] } )

for a in Gex[0] :
  for b in voisins(a,Gex) :
    if b in voisins(3,Gex) :
      print(a)
```

2 - Simplet [7 points]

Compléter la case ci-dessous avec le code Python d'une fonction `isoles(G)` donnant les sommets de `G` qui ne sont reliés à aucun autre sommet (mais peut-être à eux-même).

Écrire aussi une fonction `simple(G)` donnant un nouveau graphe ayant les mêmes sommets que `G`, ainsi que les mêmes arêtes mais une fois seulement chaque (le graphe résultat est dit simple). Par exemple : `simple(Gex)` vaut `({3, 5, 7, 9, 42}, [{42, 5}, {9, 7}, {7}, {3, 7}, {5}])`.

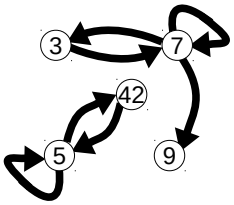
Puis écrire une fonction `noyau(G)` donnant un nouveau graphe obtenu à partir de `G` en enlevant les arêtes en double, les boucles, et les sommets isolés.

3 - Graphe Orienté [3 points]

Un graphe (orienté) simple peut être représenté en Python par un couple (V,E) où V est l'ensemble des sommets et E l'ensemble des arêtes. Les arêtes sont des couples (des tuples en Python) de sommets.

```
Gor = ( {3,7,9,5,42}, { (5,42), (7,9), (7,7), (7,3), (42,5), (5,5), (3,7) } )

def diffuseVoisins(s,GG,f) :
    res = set() # l'ensemble vide
    for t in GG[0] :
        if (s,t) in GG[1] :
            res = res | { f(s,t) } # union ensembliste
    return res
```




```
>>> def dist(a,b) :
      return abs(a-b)
>>> print(diffuseVoisins(7,Gor,dist ) )
      {0, 2, 4}
```

```
>>> print(diffuseVoisins(3,Gor,lambda x,y : y))
      {7}
```

Compléter similairement le tableau ci-dessous avec l'affichage produit par le code fourni.

```
print( (lambda a,b : a+b)(7,8) )
print( { dist(a,b) for a in Gor[0] for b in Gor[0] if (a,b) in Gor[1] } )
print( diffuseVoisins(5,Gor,lambda x,y : (y,x) ) )
```

4 - Déconnexe [7 points]

Compléter la case ci-dessous avec le code d'une fonction `cibles(s,GG)` donnant l'ensemble des sommets de GG figurant au bout d'une flèche partant de s.

Par exemple : `cibles(7,Gor)` vaut `{3,7,9}`, `cibles(9,Gor)` vaut `set()`.

Écrire aussi le code d'une fonction `access(s,GG)` donnant l'ensemble des sommets de GG accessibles à partir de `s` en suivant un nombre quelconque de flèches (toutes dans le bon sens).

Par exemple `access(3,Gor)` vaut `{3,9,7}`, et `access(9,Gor)` vaut `{9}`.

Ajouter le code d'une fonction `fortCon(GG)` disant si le graphe `GG` est fortement connexe, c'est à dire que pour tous sommets `u` et `w` de `GG`, il existe (dans `GG`) une suite de flèches successives (toujours dans le bon sens) menant de `u` à `w`.

Par exemple ({1,2,3}, { (1,2),(2,3),(3,1) }) est fortement connexe, ({4,7,9}, { (4,7),(4,9),(7,9) }) ne l'est pas