

Numéro d'anonymat :

Groupe :

Remplir l'en-tête de la copie, cacheter, et ne rien écrire d'autre sur la copie ; porter les réponses directement sur le sujet, et glisser celui-ci dans la copie en fin d'épreuve. *Ne pas oublier d'indiquer ci-dessus le numéro d'anonymat et le groupe.*

Le sujet comporte 5 exercices et 6 pages. *Aucun document n'est autorisé* : la liste des fonctions de manipulation des graphes est distribuée en annexe sur une feuille séparée.

Le sujet est long, pas de panique, le barème en tiendra compte, il n'est absolument pas nécessaire de tout traiter pour obtenir une bonne note ; ne jamais oublier qu'*une bonne réponse vaut mieux que deux mauvaises.*

**Exercice 1** On considère la fonction `mystere` suivante :

```
def mystere(x):  
    z=0  
    y=0  
    while z <= x:  
        z=z+2*y+1  
        y=y+1  
    return y-1
```

1. Simuler l'exécution de `mystere(39)` en complétant le tableau suivant :

$z$	0	1	4					
$y$	0	1	2	3	4	5	6	7

2. Que vaut `mystere(39)` ?

3. On exécute la suite de trois instructions :

```
>>> m = 2009  
>>> n = m*m  
>>> mystere(n)
```

Quel entier va être affiché par la troisième instruction ?

4. Soit  $p$  un entier quelconque, donner une formule simple pour `mystere(p*p)` :

5. En déduire une formule simple pour `mystere(x)` :

*Indication* : on pourra utiliser la fonction  $x \mapsto E(x)$  (la *partie entière* de  $x$ ) définie par :  $\forall x \in \mathbb{R}, E(x) \in \mathbb{N}$  et  $E(x) \leq x < E(x) + 1$ .

## Exercice 2

1. Ecrire une fonction `aretesToutesMarquees(G)` qui retourne *True* si toutes les arêtes du graphe  $G$  sont marquées, et *False* sinon. *Attention* : le module de manipulation des graphes *ne* comporte *pas* de fonction `listeAretes`, et toute réponse qui utiliserait d'autres fonctions que celles utilisées en TD/TP et rappelées en annexe serait considérée comme nulle.

2. On appelle respectivement  $S$  et  $A$  les ensembles de sommets et d'arêtes du graphe  $G$ , et on note respectivement  $|S|$  et  $|A|$  le nombre de sommets et le nombre d'arêtes. Pendant l'exécution de la fonction `aretesToutesMarquees(G)` quel est le nombre maximal d'appels à la fonction `estMarqueeArete`? Dans quel cas cela se produit-il?

3. Dans un graphe  $G$  un ensemble de sommets  $U$  est appelé une *couverture* de  $G$  si toute arête possède au moins une extrémité dans  $U$ . Ecrire une fonction `couverture(G,U)` qui teste si  $U$  (qu'on supposera représenté par une liste) est une couverture de  $G$  :

4. Donner une estimation aussi précise que possible de la complexité *au pire* de la fonction `couverture(G,U)` en justifiant votre réponse :

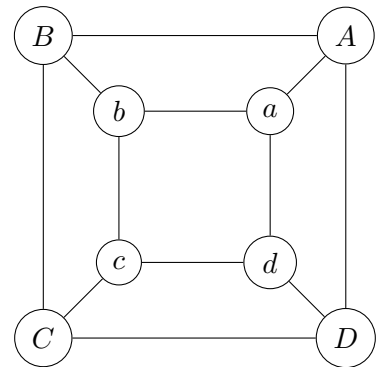
**Exercice 3** Soit  $S$  l'ensemble des sommets d'un graphe simple  $G$  et soit une bijection  $f : S \rightarrow S$  ; on dit que  $f$  est un *automorphisme* de  $G$  si :

$$\forall s \forall t \quad s \text{ et } t \text{ sont voisins dans } G \Rightarrow f(s) \text{ et } f(t) \text{ sont voisins dans } G. \quad (1)$$

1. On suppose la fonction  $f(s)$  écrite par ailleurs, écrire une fonction `automorphisme(G)` qui retourne *True* si la condition (1) ci-dessus est vérifiée, *False* sinon.

2. On suppose que  $G$  est le graphe ci-contre et que  $f(A) = A, f(B) = a, f(D) = B$ . Montrer qu'il existe un automorphisme  $f$  et un seul compatible avec ces trois valeurs, en complétant le tableau suivant (attention à écrire les lettres  $c$  et  $C$  de telle sorte que le lecteur les distingue clairement) :

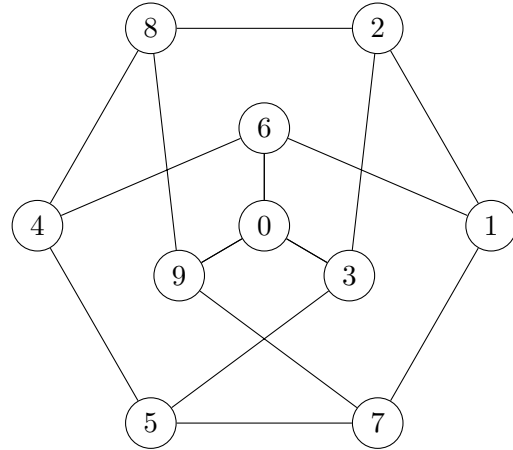
$s$	$A$	$B$	$C$	$D$	$a$	$b$	$c$	$d$
$f(s)$	$A$	$a$		$B$				



Indiquez comment vous avez raisonné pour arriver à ce résultat :

#### Exercice 4

Dans cet exercice  $P$  désigne le graphe ci-contre. Les dix sommets sont numérotés de 0 à 9. L'arête qui relie les sommets  $i$  et  $j$  est notée  $a_{ij}$  : par exemple l'arête qui relie les sommets 3 et 5 est notée  $a_{35}$  (ou bien  $a_{53}$  car l'ordre des extrémités est sans importance). On suppose que `listeSommets(P)` retourne les dix sommets du graphe  $P$  dans l'ordre de leurs numéros, et que pour chacun d'entre eux la liste des arêtes incidentes est ordonnée selon les numéros des extrémités ; par exemple si  $s_3$  désigne le sommet numéro 3, `listeAretesIncidentes(s3)` retourne la liste  $[a_{30}, a_{32}, a_{35}]$ .



On dispose des fonctions suivantes :

```
def sommetAccessible(G):
    for s in listeSommets(G):
        if not estMarqueSommet(s):
            for a in listeAretesIncidentes(s):
                t = voisinPar(s,a)
                # l'arête a relie les sommets s et t
                if estMarqueSommet(t):
                    marquerArete(a)
                    marquerSommet(s)
            return s
    return None

def marquerArbre(G):
    s = sommetNumero(G,0)
    marquerSommet(s)
    while s != None:
        s = sommetAccessible(G)
```

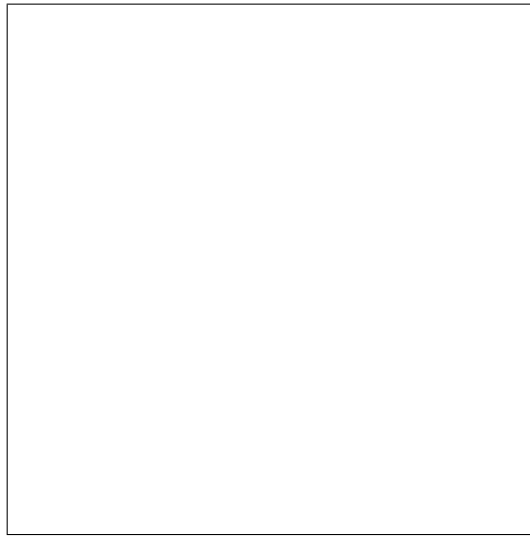
On simule l'exécution de l'instruction `marquerArbre(P)` (au départ aucun sommet ni aucune arête ne sont marqués).

1. La fonction `marquerArbre` commence par marquer le sommet initial 0, puis appelle une première fois la fonction `sommetAccessible`, qui marque l'arête  $a_{30}$  et le sommet 3 ; compléter ci-dessous la liste des arêtes et des sommets marqués *dans l'ordre* des opérations de marquage :

0  $a_{30}$  3 ...

*Attention* : simuler calmement et soigneusement l'exécution du programme, il y a une seule réponse correcte.

2. Dessiner le graphe  $T$  formé des sommets et des arêtes marqués (placer impérativement les sommets comme sur le dessin du graphe  $P$ ) :



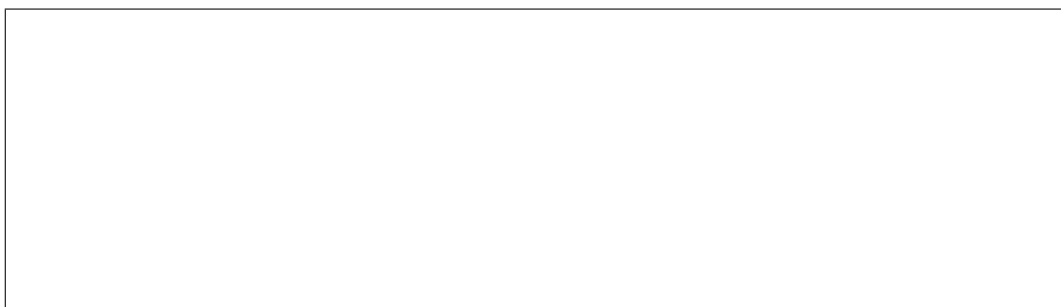
3. Pourquoi le graphe  $T$  est-il un arbre ?



4. Pour chaque arête  $a$  non marquée de  $P$  (c'est-à-dire pour chaque arête  $a$  qui n'appartient pas à  $T$ ) donner un cycle simple  $\Gamma_a$  de  $P$  qui contienne  $a$  et dont toutes les autres arêtes soient marquées (c'est-à-dire appartiennent à  $T$ ) ; comme le graphe est simple on pourra désigner un cycle par la suite de ses sommets, en omettant les arêtes :



Pour une arête  $a$  non marquée existe-t-il plusieurs choix pour le cycle  $\Gamma_a$  ? Pour rapporter des points la réponse doit être justifiée :



**Exercice 5** Le graphe *circulaire*  $C_n$  est composé d'un seul cycle de  $n$  sommets ; par exemple  $C_3$  est un triangle,  $C_4$  est un carré,  $C_5$  est un pentagone, etc. Si  $[s_1, s_2, \dots, s_n]$  est la liste retournée par la fonction `listeSommets(G)` lorsque  $G$  est le graphe circulaire  $C_n$ , celui-ci est constitué du cycle  $s_1 - s_2 - \dots - s_n - s_1$ .

Ecrire une fonction `coloriageCirculaire(G,c1,c2,c3)` qui colorie un graphe circulaire avec trois couleurs  $c_1, c_2, c_3$  en prenant garde de n'utiliser que deux couleurs lorsque c'est possible (on suppose que  $G$  est bien un graphe circulaire, on ne demande pas d'écrire le code pour le vérifier — sauf si vous vous ennuyez) :



## Annexe

Liste des fonctions disponibles pour manipuler les graphes ; cette feuille n'est pas à rendre avec le devoir, ne rien écrire dessus.

L'argument $G$ est un graphe	
<code>listeSommets(G)</code>	retourne la <i>liste</i> des <i>sommets</i> de $G$ .
<code>nbSommets(G)</code>	retourne le <i>nombre</i> de <i>sommets</i> de $G$ .
<code>sommetNom(G,etiquette)</code>	retourne le <i>sommet</i> de $G$ désigné par son <i>nom</i> (étiquette). Exemple : <code>sommetNom (Europe, 'Italie')</code> .
<code>sommetNumero(G,i)</code>	retourne le <i>sommet</i> numéro $i$ dans $G$ ; la numérotation commence à 0.

L'argument $s$ est un sommet	
<code>listeVoisins(s)</code>	retourne la <i>liste</i> des <i>voisins</i> de $s$ .
<code>degre(s)</code>	retourne le <i>degré</i> de $s$ .
<code>nomSommet(s)</code>	retourne le <i>nom</i> (étiquette) de $s$ .
<code>colorierSommet(s,c)</code>	colorie $s$ avec la couleur $c$ . Exemples de couleurs : 'red', 'green', 'blue', None.
<code>couleurSommet(s)</code>	retourne la <i>couleur</i> de $s$ .
<code>marquerSommet(s)</code> <code>demarquerSommet(s)</code>	marque ou démarque $s$ .
<code>estMarqueSommet(s)</code>	retourne <code>True</code> si $s$ est marqué, <code>False</code> sinon.
<code>listeAretesIncidentes(s)</code>	retourne la <i>liste</i> des arêtes <i>incidentes</i> à $s$ .
<code>voisinPar(s,a)</code>	retourne le voisin de $s$ en suivant l'arête $a$ .

L'argument $a$ est une arête	
<code>nomArete(a)</code>	retourne le <i>nom</i> (étiquette) de $a$ .
<code>marquerArete(a)</code> <code>demarquerArete(a)</code>	marque ou démarque $a$ .
<code>estMarqueeArete(a)</code>	retourne <code>True</code> si $a$ est marquée, <code>False</code> sinon.