Les exercices peuvent être résolus indépendamment. Indiquez votre groupe sur votre copie.

1 Code mystère

```
@trace
def mystere(s, acc):
    if s == "":
        res = acc
    else:
        res = mystère(s[1:], s[0] + acc)
    return res
```

1. Tracez les appels de `mystere` lors du calcul de `mystere("cursif", "er")`
2. Rédigez la documentation de la fonction.
3. Donnez un variant possible $v(s, acc)$ de `mystere`. On rappelle qu'un variant est une expression de type entier, positive, utilisant uniquement les paramètres de la fonction, strictement décroissante lors des appels récursifs.

2 PGCD binaire

Pour calculer le plus grand commun diviseur (PGCD) de deux entiers a et b non tous-nuls, il est possible d'utiliser les formules suivantes valables si $a \geqslant b$:

Condition	Formule
$b = 0$	$pgcd(a, 0) = a$
a et b sont pairs	$pgcd(a, b) = 2 \times pgcd(\frac{a}{2}, \frac{b}{2})$
a est pair et b impair	$pgcd(a, b) = pgcd(\frac{a}{2}, b)$
a est impair et b pair	$pgcd(a, b) = pgcd(a, \frac{b}{2})$
a et b sont impairs	$pgcd(a, b) = pgcd(\frac{a-b}{2}, b)$

Si $a < b$, on remarque que $pgcd(a, b) = pgcd(b, a)$

4. Utilisez ces propriétés pour calculer le pgcd de 60 et 42.
5. Rédigez une fonction récursive `euclide_binaire` prenant en paramètre deux entiers naturels `a` et `b` et qui renvoie le pgcd de `a` et de `b`, calculé en utilisant ces propriétés.

3 Contacts

Un étudiant cherche à écrire une application de gestion de contacts en Python. Pour gérer les contacts, il décide d'utiliser une liste dont les éléments sont appelés fiches.

Chaque fiche de contact sera représentée par un dictionnaire. Les clés de ce dictionnaire sont des chaînes de caractères, ainsi que les valeurs associées. Pour qu'un dictionnaire soit une fiche de contact, il faut qu'il possède au moins deux associations :

- à la clé `"nom"` doit être associé le nom de la personne ;
- à la clé `"number"` doit être associé le numéro de téléphone de la personne.

et que dans ces deux associations, les valeurs aient le type `str`

6. Écrire un prédicat `est_fiche` prenant en paramètre un dictionnaire `dico` et qui renvoie `True` si `dico` est une fiche de contact valide et `False` dans le cas contraire.

```
>>> est_fiche({'prenom':'Raymond', 'nom':'Calbuth', 'number':'21541212'})
True
>>> est_fiche({'nom':'Timoléon'}) # pas de clé 'number'
False
>>> est_fiche({'nom':'Usul', 'number':3615}) # 3615 est de type int, pas str
False
```

7. Rédigez un prédicat `est_liste_contact` prenant en paramètre une liste de dictionnaires et qui renvoie `True` si cette liste est une liste de fiches, et `False` dans le cas contraire. On pourra utiliser le prédicat `all` de Python.

```
>>> ex1 = [{'prenom':'Raymond', 'nom':'Calbuth', 'number':'21541212'},
...        {'nom':'messagerie', 'number':'888'}]
>>> est_liste_contact(ex1)
True
>>> est_liste_contact(ex1 + [{'nom':'Timoléon'}])
False
```

8. Rédigez une fonction `get_num` prenant en paramètre une liste `contacts` de fiches, ainsi qu'une chaîne de caractères `nom` et qui renvoie :

- le numéro de téléphone de la dernière personne de la liste ayant pour nom `nom` s'il y en a au moins une ;
- Une chaîne vide si `nom` n'est pas le nom d'une personne de la liste.

```
>>> ex1 = [{'prenom':'Raymond', 'nom':'Calbuth', 'number':'21541212'},
...        {'nom':'messagerie', 'number':'888'}]
>>> get_num(ex1, 'messagerie')
'888'
>>> get_num(ex1, 'Haddock')
''
```

9. On suppose qu'un numéro de téléphone n'est attribué qu'à une seule personne.

Rédigez une fonction `annuaire_inverse` prenant en paramètre une liste `contacts` de fiches, et qui renvoie un dictionnaire dont les clés sont les numéros de téléphone (sous forme de chaînes de caractères) et les valeurs associées le nom de la personne possédant ce numéro.

```
>>> ex1 = [{'prenom':'Raymond', 'nom':'Calbuth', 'number':'21541212'},
...        {'nom':'messagerie', 'number':'888'}]
>>> annuaire_inverse(ex1) == {'21541212':'Calbuth', '888':'messagerie'},
True
```

10. Définissez un prédicat `a_des_numeros_dupliques` paramétré par une liste de fiches et renvoie `True` si cette liste contient au moins deux fiches avec le même numéro, et `False` sinon.

```
>>> ex1 = [{'prenom':'Raymond', 'nom':'Calbuth', 'number':'21541212'},
...        {'nom':'messagerie', 'number':'888'}]
>>> ex2 = [{'nom':'Haddock', 'number':'+32 (0)2 626 24 21'},
           {'nom':'Sanzot', 'number':'+32 (0)2 626 24 21'}]
>>> a_des_numeros_dupliques(ex1)
False
>>> a_des_numeros_dupliques(ex2)
True
```

4 Recherche dichotomique

Soit un tableau `t` d'entiers représenté ci-dessous.

t :	2	7	24	32	41	45	51	59	62	66	71	74	75	81	97	99

11. Pourquoi est-il possible d'appliquer l'algorithme de recherche dichotomique sur ce tableau ?

12. Donnez la suite des tranches de `t` parcourue lors de la recherche dichotomique de 42 dans tout le tableau `t`. Vous pouvez décrire cette suite, au choix :

- soit en présentant sous forme de tableau les valeurs prises par les indices `d` et `f` à la fin de chacune des itérations de la boucle de la version itérative ;
- soit en traçant les appels de la version récursive.