Math93.com

TD - NSI
Minis Projets Python

Les mini projets sont des épreuves de qualification pour le concours PROLOGIN adaptées à pyhton

Projet I
Prologin 2019

1 Arnaque aérienne - Niveau 1

1.1 Énoncé

Comme le disait le grand-père de Joseph Marchand, « Le plus beau voyage est celui qu'on n'a pas encore fait ». New York était dans la tête de Joseph depuis plusieurs années maintenant, et il a décidé aujourd'hui d'acheter son billet d'avion.
Quelques secondes avant de cliquer sur le bouton « Acheter! », son amie Haruhi lui envoie une liste de prix qu'elle a trouvés sur Internet. Joseph est curieux de voir si ces derniers sont moins chers que le billet qu'il s'apprêtait à acheter.

1.2 Entrée

— Le prix initial du billet de Joseph.

— Une liste contenant les N prix trouvés par Haruhi..

1.3 Sortie

Si Haruhi a trouvé au moins 3 prix strictement moins chers que celui de Joseph, affichez « ARNAQUE! » pour l'avertir. Sinon « Ok bon voyage, bisous, n'oublie pas de m'envoyer des photos! ».

1.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>>  prixInit=570
>>>  listeHaruhi= [495, 1200 ,540 ,450]
>>>  arnaqueAerienne(prixInit,listeHaruhi)
ARNAQUE
```

Commentaire Exactement 3 billets sont strictement moins chers que celui choisi par Joseph : 495, 540, et 450. Ça sent l'arnaque...

```
# Dans la console Python
>>>  prixInit=820
>>>  listeHaruhi= [580, 2000, 970, 1050, 820]
>>>  arnaqueAerienne(prixInit,listeHaruhi)
Ok bon voyage, bisous, n'oublie pas de m'envoyer des photos !
```

Commentaire Seul le billet à 580€ est strictement inférieur au billet de Joseph à 820€.

1.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

2 Méli-mélo temporel – Niveau 2

2.1 Énoncé

Joseph et Haruhi sont maintenant partis dans leurs voyages endiablés. Ils ne voyagent pas dans les mêmes pays et sont donc sur des fuseaux horaires différents. Pour toujours savoir combien de décalage ils ont entre eux, ils ont noté le décalage de tous les pays par rapport à la France. Maintenant à chaque fois que l'un ou l'autre arrive dans un pays, ils veulent prévenir l'autre du décalage entre eux.

2.2 Entrée

L'entrée comprendra :

— Une liste d'entier relatif d_i le décalage entre la France et le pays d'indice i

— Une liste de couple contenant les entiers 1 ou 2 pour indiquer un déplacement de Haruhi ou Joseph et un entier positif indiquant l'indice du pays.

2.3 Sortie

Pour chaque déplacement de Haruhi ou Joseph, vous afficherez leurs décalages horaire en valeur absolue dans une liste.

2.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>>  decalPays=[12,-6,-2]
>>>  voyage= [(1,1),(2,0),(1,2)]
>>>  meliMelotemp(decalPays,voyage)
[6,18,14]
```

Commentaire Haruhi part en premier, elle a 6h de décalage part rapport à la France donc à Joseph. Puis Joseph part et à 12h de plus donc ils ont 18h de décalage, puis Haruhi se rapproche donc ils reviennent à 14h de décalage.

```
# Dans la console Python
>>>  decalPays=[5,-12,12]
>>>  voyage= [(2,1),(1,2)]
>>>  meliMelotemp(decalPays,voyage)
[6,18,14]
```

Commentaire Joseph Marchand arrive à Pago Pago, 12 heures de décalage, puis Haruhi arrive à Atafu, ils ont désormais 24h de décalage bien qu'ils ne soient qu'à plus ou moins 600km!

2.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

3 Manhattan maboul –Niveau 3

3.1 Énoncé

Ah Manhattan... Le quartier préféré de Haruhi! Nombreux de ses amis sont d'ailleurs de passage à New-York et Haruhi aimerait les revoir depuis son dernier long voyage.Adepte des finales Prologin, Haruhi a pour objectif de rencontrer en M jours consécutifs un nombre maximal de ses amis. Pour cela elle a noté l'arrivée de tout le monde. À noter qu'il est possible que plusieurs amis soient présents un même jour.

3.2 Entrée

— Un nombre M correspondant au nombre de jours consécutifs que Haruhi peut utiliser pour rencontrer ses amis.

— Une liste de N entiers, représentant un jour où un des amis de Haruhi est à Manhattan.

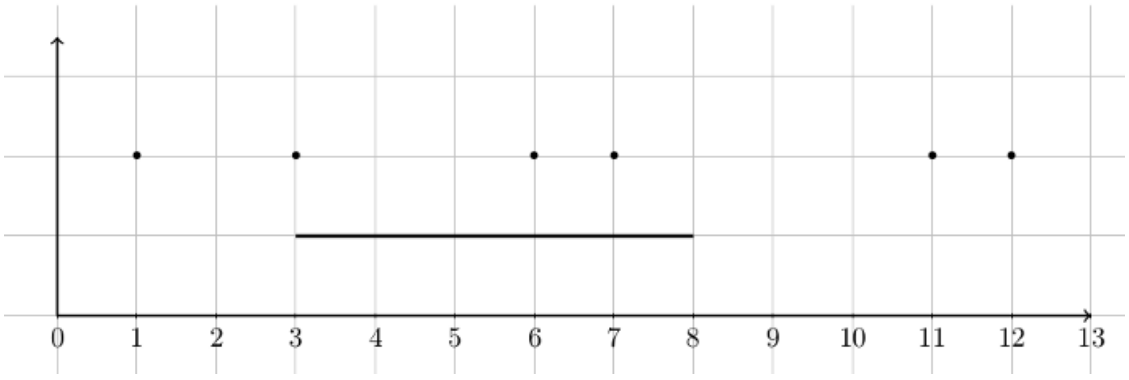
3.3 Sortie

Le nombre d'amis maximum que Haruhi peut revoir en M jours consécutifs

3.4 Exemples d'entrée/sortie

Exemple 1

```
# Dans la console Python
>>>  jourConsec=5
>>>  jourAmis= [3, 11, 1, 7, 6, 12]
>>>  manhathanMaboul(jourConsec,jourAmis)
3
```

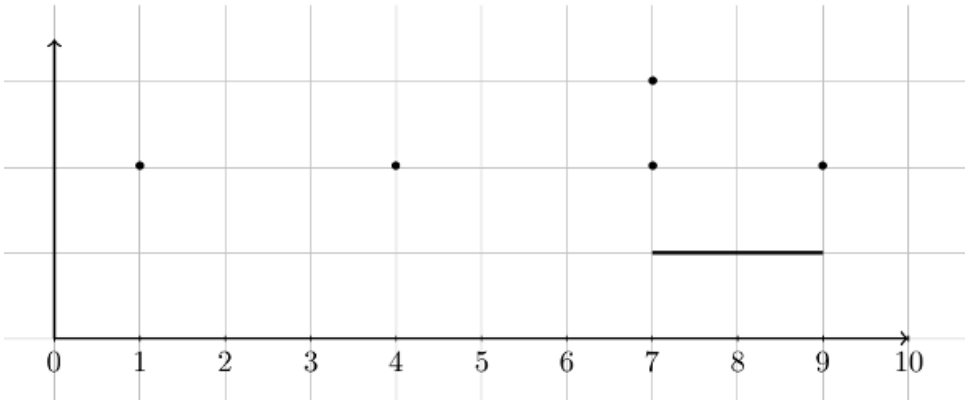


Commentaire

Haruhi peut rencontrer 3 de ses amis en 5 jours consécutifs.

Exemple 2

```
# Dans la console Python
>>>  jourConsec=2
>>>  jourAmis= [4,1,7,9,7]
>>>  manhathanMaboul(jourConsec,jourAmis)
3
```



Commentaire

Dans le meilleur des cas, Haruhi peut croiser 3 de ses amis en 2 jours consécutifs en sélectionnant les 3 derniers de la liste.

Projet II

Prologin 2011

4 Les « 101 » Dalmatiens - Partie I – Niveau 1

4.1 Énoncé

Joseph Marchand veut faire une farce à son fidèle dalmatien Scooby-Naire : il souhaite accrocher dans sa niche une photo de ce bel animal, en inversant au préalable les couleurs blanc et noir.
On vous donne un tableau de bits représentant une image en noir et blanc, vous devez implémenter une fonction qui calcule le négatif de cette image.

4.2 Entrée

Le tableau de bits (N x M) fait avec une liste de liste, par exemple pour un tableau 3x5 :

```
#dans l'éditeur PYTHON
exemple=[
[0,0,0,1,0],
[1,1,1,1,1],
[1,0,1,0,1]]
```

4.3 Sortie

Le négatif de l'image (N x M) fait avec une liste de liste.

4.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
monChien=[
[0,0,0],
[1,0,1],
[0,1,0]]
```

```
# Dans la console Python
>>>  dalmatienI(monChien)
[[1,1,1], [0,1,0],[1,0,1]]
```

4.5 Rendu

— Un code python dans le bon format.
— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

5 Les « 101 » Dalmatiens - Partie II –Niveau 2

5.1 Énoncé

Joseph Marchand, vous l'aurez compris, a des distractions simples. Il aime particulièrement imprimer des photos de Scooby-Naire sur papier calque et s'amuser à les retourner horizontalement pendant des heures.

Cependant, il peut arriver que les deux faces du calque soient identiques (auquel cas, le jeu perd tout son intérêt). Pour éviter tout effort inutile, Joseph vous demande de l'aider.
On vous donne un tableau de bits représentant une image en noir et blanc, vous devez dire si cette image admet un axe de symétrie vertical au milieu de l'image.

5.2 Entrée

Le tableau de bits (N x M) fait avec une liste de liste.

5.3 Sortie

True si l'image admet un axe de symétrie vertical au milieu de l'image, False sinon.

5.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
monChien=[
[0, 0, 1, 1, 0, 0],
[0, 1, 1, 1, 1, 0],
[0, 0, 1, 1, 0, 0]]
```

```
# Dans la console Python
>>>  dalmatienII(monChien)
True
```

6 Pince os – Niveau 3

6.1 Énoncé

Aujourd'hui, Joseph Marchand se sent l'âme d'un artiste. Il va peindre sa meilleure nature morte : Scooby-Naire.
On vous donne un tableau de bits représentant une image en noir et blanc, ProloGIMP 0.4.2 ne dispose que d'un pinceau en forme de " + " (3 x 3 pixels). Chaque application du pinceau sur l'image (le centre du pinceau devant être dans l'image, sinon le logiciel plante) noircit les 5 pixels affectés. Vous devez implémenter une fonction déterminant s'il est possible de dessiner l'image donnée en entrée avec le pinceau, en partant d'une image dont tous les pixels sont blancs.

6.2 Entrée

Le tableau de pixels (N x M) fait avec une liste de liste, 0 représente la couleur blanche, et 1 la couleur noire.

6.3 Sortie

True si l'image est dessinable avec le pinceau, False sinon.

6.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
monImage1=[
[1, 1, 1, 0, 0],
[1, 1, 1, 1, 0],
[0, 1, 1, 0, 0]]
```

```
monImage2=[
[1, 0, 0, 1, 1, 1],
[1, 1, 0, 1, 1, 1],
[1, 0, 0, 0, 1, 0]]
```

```
# Dans la console Python
>>>  pinceOs(monImage1)
True
>>>  pinceOs(monImage2)
False
```

6.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

7 Corruption – Niveau 4

7.1 Énoncé

La licence d'utilisation de ProloGIMP a expiré. Joseph Marchand aimant autant l'argent que son chien, il refuse de la renouveler et décide plutôt de modifier des bits au hasard du fichier binaire de l'application. Grave erreur, il corrompt le pinceau !
On vous donne un tableau de bits représentant une image en noir et blanc, vous disposez d'un pinceau en forme de « + » (3 * 3 pixels). Chaque application du pinceau sur l'image (le centre du pinceau devant être dans l'image) inverse la couleur (les bits) des 5 pixels affectés. Vous devez implémenter une fonction déterminant s'il est possible de dessiner l'image donnée en entrée avec le pinceau, en partant d'une image dont tous les pixels sont blancs (à 0).
Dans le premier exemple, il est possible d'obtenir l'image en appliquant deux fois le pinceau : une première fois à la position (ligne, colonne) = (1, 1) et la deuxieme fois à la position (1, 2).

7.2 Entrée

Le tableau de pixels (N x M) fait avec une liste de liste, 0 représente la couleur blanche, et 1 la couleur noire.

7.3 Sortie

True si l'image est dessinable avec le pinceau, False sinon.

7.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
monImage1=[
[0, 1, 1],
[1, 0, 0],
[0, 1, 1]]
monImage2=[
[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
```

```
        [0, 1, 0, 0, 0],
        [0, 0, 0, 0, 0]]
        monImage3=[
        [0, 1, 0],
        [1, 1, 1],
        [0, 0, 0],
        [1, 1, 1],
        [0, 0, 0],
        [1, 0, 1],
        [1, 0, 1],
        [0, 1, 0],
        [0, 0, 0],
        [0, 0, 0]]
```

```
# Dans la console Python
>>>  corruption(monImage1)
True
>>>  corruption(monImage2)
False
>>> corruption(monImage3)
True
```

7.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

Projet III

PROLOGIN 2010

8 Nucléotide – Niveau 1

8.1 Énoncé

Une séquence d'ADN sera une suite finie constituée de lettres dans l'ensemble A, T, G, C. On vous donne en entrée une séquence d'ADN de longueur N. Écrivez une fonction qui renvoie le nucléotide (la lettre) le plus présent. Si c'est le cas de plusieurs, renvoyez celui qui vient en premier dans l'ordre alphabétique.

8.2 Entrée

Une chaîne de caractères, la séquence d'ADN de longueur N.

8.3 Sortie

Le nucléotide le plus fréquent dans la séquence d'ADN.

8.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>>  chaineADN="ATTGCCATATCC"
C
>>>  chaineADN="AAAACCCGGGTTT"
A
```

8.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

9 Les acides aminés -Niveau 2

9.1 Énoncé

Une séquence d'ADN sera une suite finie constituée de lettres dans l'ensemble A, T, G, C. Lors de la traduction (simplifiée!) d'une séquence d'ADN en suite d'acides aminés, chaque groupement de trois nucléotides de la séquence d'ADN est transformé en acide aminé. Écrivez une fonction qui, étant donné une séquence d'ADN de longueur N et la table de traduction de taille M, renvoie la suite d'acides aminés correspondante. On assure que tout le brin d'ADN pourra être traduit, que N est multiple de 3, et que la table de traduction est correcte.

9.2 Entrée

— Un dictionnaire qui associe trois nucléotides (en lettres capitales) la traduction correspondante.

— Une chaîne de caractères, la séquence d'ADN de longueur N.

9.3 Sortie

La liste d'acides aminés traduits depuis la séquence d'ADN grâce au dictionnaire de traduction.

9.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
monDicoADN1={"ATT":"isoleucine", "TCC":"serine","GCC":"alanine"}

monDicoADN2={"TTT": "phenylalanine","TTC": "phenylalanine",
             "TTA": "leucine","TTG": "leucine","TCT": "serine",
             "TCC": "serine","TCA": "serine","TCG": "serine",
             "TAT"; "tyrosine","TAC": "tyrosine","TGT": "cysteine",
             "TGC": "cysteine","TGG": "tryptophane","CTT": "leucine",
             "CTC": "leucine","CTA": "leucine","CTG": "leucine",
             "GTT": "valine"}

ADN1="ATTGCCTCC"
ADN2="TTTTGGCTTCTCCTATGTTACTCGTTAGTT"
```

```
# Dans la console Python
>>>  acideAmineListe(monDicoADN1,ADN1)
[soleucine, alanine, serine]
>>>  acideAmineListe(monDicoADN2,ADN2)
[phenylalanine, tryptophane, leucine, leucine, leucine, cysteine,
tyrosine, serine, leucine, valine]
```

9.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

10 Sous-séquence –Niveau 3

10.1 Énoncé

Une séquence d'ADN sera une suite finie constituée de lettres dans l'ensemble A, T, G, C. On cherche à analyser les fréquences d'apparition des sous-séquences d'une séquence d'ADN de longueur N donnée en entrée. Ecrivez une fonction qui renvoie la sous-séquence contiguë de longueur L de la chaîne d'ADN la plus fréquente. Dans le cas où plusieurs sous-séquences de longueur L apparaissent un même nombre de fois, affichez celle qui vient en premier dans l'ordre alphabétique.

10.2 Entrée

— Un entier L.

— Une chaîne de caractères, la séquence d'ADN de longueur N.

10.3 Sortie

La sous-séquence d'ADN de longueur L recherchée.

10.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>> N=2
>>> ADN="TCGTACGTAG"
>>> sousSequence(N,ADN)
"CG"
>>> N=4
>>> ADN="AATTCGGCCGATCGTCGAATTCGATA"
>>> sousSequence(N,ADN)
"AATT"
```

10.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

11 Timestamps – Niveau 1

11.1 Énoncé

En épreuve machine de demi-finale, vous soumettrez des solutions pour divers problèmes. On vous donne six variables entières représentant trois heures de soumission (données sous la forme de deux entiers, l'un pour les heures et l'autre pour les minutes). Écrivez une fonction qui détermine l'heure de la première soumission. Les trois heures données à votre programme seront toujours différentes.

11.2 Entrée

Une liste de trois couples d'entier, le premier étant l'heure de soumission et le second, les minutes. Sur chaque ligne, les deux entiers seront séparés par un espace.

11.3 Sortie

Renvoyez 1, 2, ou 3, si la première, la deuxième, ou la troisième heure est la première soumission.

11.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>> soumission=[(10,48),(9,3),(9,39)]
>>> timeStamps(soumission)
2
>>> soumission=[(7,58),(20,26),(9,29)]
>>> timeStamps(soumission)
1
```

11.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

12 GPS – Niveau 2

12.1 Énoncé

On vous donne une liste de coordonnées de type (x_i, y_i) , nombres entiers, représentant les coordonnées cartésiennes sur une carte de France des différents centres d'examen pour les demi-finales. Vous vous situez en (x, y). Ecrivez une fonction qui renvoie le centre le plus proche de vous. Vous utiliserez la distance euclidienne dans vos calculs.

12.2 Entrée

— Un couple d'entier (X, Y), indiquant votre positions sur la carte.

— Une liste de coordonnées (x_i, y_i) , représentant les coordonnées des centres d'examen.

12.3 Sortie

Un couple contenant deux entiers, représentant les coordonnées du centre le plus proche de vous. Dans le cas où plusieurs centres sont à la même distance de vous, renvoyez le premier apparaissant dans l'entrée.

12.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>> mesCoords=(38,5)
>>> centreExam=[(54, 82),(75, 21),( 6, 21),( 61, 21),( 60, 68)]
>>> GPS(mesCoords,centreExam)
(61,21)
```

12.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

13 Le fil d'Ariane – Niveau 3

13.1 Énoncé

TTY, la chatte mythique de Prologin, est perdue dans le campus de Polytechnique. Celui-ci est donné sous la forme d'un labyrinthe rectangulaire de H*L cases, où une case est soit vide, soit un mur infranchissable. Une gamelle se trouve dans le labyrinthe. Votre tâche est de rassurer, le cas échéant, TTY : pourra-t-elle ou non atteindre sa gamelle? Vous devez donc écrire un programme qui, étant donné la position de TTY, la position de sa gamelle, et le labyrinthe, renvoie si elle peut ou non atteindre sa gamelle.

13.2 Entrée

— Un couple de deux entiers : (ytty,xtty) , la position de départ de tty (la ligne, puis la colonne).

— Un couple de deux entiers : (ygam,xgam), la position de la gamelle (la ligne, puis la colonne).

— Un labyrinthe représenté par un tableau (NxM) avec une liste de liste de 0 et de 1 :0 ou 1, représentant respectivement une case vide et un mur infranchissable. Note : La case (0,0) est située en haut à gauche. TTY ne peut se déplacer que verticalement et horizontalement.

13.3 Sortie

Un entier, 1 ou 0, indiquant si oui ou non TTY peut atteindre sa gamelle.

13.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
TTY=(0,0)
GAM=(3,0)
labyrinthe1=[
[0,0,0,1],
[1,1,0,0],
[0,0,0,0],
[0,1,1,1]]
labyrinthe2=[
[0,0,0,1],
[1,1,1,1],
[0,0,0,0],
[0,1,1,1]]
```

```
# Dans la console Python
>>>  filArianne(TTY,GAM,labyrinthe1)
True
>>>  filArianne(TTY,GAM,labyrinthe2)
False
```

13.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

Projet IV

Prologin 2008

14 Les assiettes – Niveau 1

14.1 Énoncé

Joseph Marchand est un homme très ordonné et il souhaite ranger ses assiettes en trois piles égales. Écrivez une fonction qui prend en argument trois entiers représentant respectivement la hauteur des trois piles actuelles et qui retourne 1 s'il est possible d'obtenir trois piles égales, 0 sinon.

14.2 Entrée

L'entrée compte 3 entiers, N, M et P, les hauteurs des trois piles d'assiette.

14.3 Sortie

La sortie contiendra True s'il est possible de rééquilibrer les piles d'assiette, False sinon. Contraintes d'exécution

14.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>> N,M,P=1,2,3
>>> lesAssiettes(N,M,P)
True
>>> N,M,P=0,0,2
>>> lesAssiettes(N,M,P)
False
```

14.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

15 Erreur de frappe – Niveau 2

15.1 Énoncé

Joseph Marchand est également un incorrigible romantique et il a écrit un poème pour son amie. Inattentif, il n'a pas remarqué que la disposition du clavier avait changé. On vous donne la disposition du clavier sur lequel il a tapé, sur lequel il pensait taper et le message écrit. Écrivez une fonction qui affiche le message qu'il a voulu écrire.

15.2 Entrée

— Une chaîne de caractères représentant les touches erronées du clavier sur lequel Joseph Marchand pensait taper, dans un ordre arbitraire.

— une chaîne de caractères représentant les touches réelles du clavier sur lequel Joseph Marchand a effectivement tapé, dans le même ordre que la première chaine.

— Une chaîne de caractère qui représente le texte qui s'est affiché sur l'écran de Joseph Marchand. Ce texte ne contiendra pas d'accents,

15.3 Sortie

Une chaine de caractére qui représente le texte que Joseph Marchand voulait taper.

15.4 Commentaires

Pour plus de simplicités, les deux claviers ne différeront que par l'emplacement des lettres. Ainsi, les deux chaînes représentant les claviers ne contiendront que des lettres majuscules, et les lettres du textes d'entrée seront également en majuscule (par contre, le texte d'entrée peut contenir des symboles de ponctuation).

15.5 Exemples d'entrée/sortie

```
# Dans la console Python
>>> toucheError="QZAW"
>>> toucheClavi="AWQZ"
>>> texte="LE ZQGON EST HQWQRDEUX !"
>>> erreurFrappe(toucheError,toucheClavi,texte)
"LE WAGON EST HAZARDEUX !"
>>> toucheError="BAT"
>>> toucheClavi="ATB"
>>> texte="LES ATBETUX SONB AETUX"
>>> erreurFrappe(toucheError,toucheClavi,texte)
"LES BATEAUX SONT BEAUX"
```

15.6 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

16 Les prisonniers – Niveau 3

16.1 Énoncé

Dans une mise en scène imaginaire, réalisée pour les seuls besoins de l'exercice, les N gardes du couloir de N cellules d'une prison dans lesquelles sont enfermés N prisonniers se livrent à un jeu étrange avant de partir le soir. Chacun, tour à tour, va changer l'état des portes de certaines cellules : s'il trouve une porte ouverte, il la ferme, et réciproquement s'il trouve une porte fermée, il l'ouvre. Le premier garde change l'état de toutes les portes, le deuxième change l'état d'une porte sur deux (c'est-à-dire des portes 2, 4, 6, 8...), le troisième une porte sur trois (3, 6, 9...). Ce processus se répète jusqu'au dernier garde. Écrivez une fonction qui prend en argument le nombre de gardes (et donc de cellules et de prisonniers) N et un numéro de prisonnier P, et qui renvoie True s'il pourra s'échapper le lendemain, False sinon.

16.2 Entrée

Deux entiers :

— N, le nombre de gardes (qui est égal au nombre de prisonniers)

— P, le numéro du prisonnier considéré. $1 \leq P \leq N$

16.3 Sortie

Un booléen :

— False si le prisonnier reste enfermé

— True si le prisonnier peut sortir

16.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>> N,P=10,1
>>> lesPrisonniers(N,P)
True
>>> N,P=100,36
>>> lesPrisonniers(N,P)
True
```

16.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

17 Mots mêlés – Niveau 4

17.1 Énoncé

Vous connaissez certainement ce jeu classique : les mots mêlés. Pour rappel, il s'agit de trouver des mots disposés horizontalement ou verticalement dans une grille remplie de lettres. Les informaticiens étant d'infatigables paresseux, il serait pratique d'avoir un programme qui fait la recherche à votre place. Vous devez donc écrire un programme qui, étant donnée une grille de N par M cases en entrée et d'un dictionnaire de P mots, renvoie le nombre de mots trouvés dans la grille.

17.2 Entrée

— La liste des mots à chercher dans la grille. Chaque mot contiendra entre 1 et 500 lettres capitales ASCII.

— la grille (NxM) représentée par une liste de chaîne. Chacune des lignes de la grille est une chaîne de M lettres capitales ASCII.

17.3 Sortie

Un entier, indiquant le nombre de mots du "dictionnaire" présents dans la grille.

17.4 Commentaires

— La grille, ainsi que les mots à chercher ne comporteront que des lettres capitales.

— Un mot qui est trouvé plusieurs fois ne doit être compté qu'une fois.

— Tous les mots de la liste sont différents.

— Chaque lettre de la grille peut être utilisée sans limitation.

— Les mots peuvent être lus dans la grille verticalement ou horizontalement, à l'endroit ou à l'envers.

17.5 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
listeMot1=["RAT","MANGER","BAZAR","CODER"]
grille1=[
"BAZAR",
"ARTRG",
"ZAMAN",
"ATYNQ",
"REDOC"
]
listeMot2=["PROLOGIN","LOGIN","UNIX","LINUX","FREEBSD","WINDOWS",
"CAML","C","PASCAL","JAVA"]
grille2=[
"ABETXINWAMDJX",
"UNILWNINRHREZ",
"GILDNIGOLQUOP",
"FXKNXGOISNXAZ",
"ENNIGOLUDNTUD",
"AICAMLATBFCQS",
"TLSBLOGINEAXB",
"RWDSBRIJACMLE",
"IILELPASCALOE",
"GNSWINDOWVUER",
"EBRDNMEPOKSDF",
"ALBLUQDSBJDOV"]
```

```
# Dans la console Python
>>>  motsMeles(listeMot1,grille1)
3
>>>  motsMeles(listeMot2,grille21)
6
```

17.6 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

Projet V
Prologin 2007

18 Bissextile – Niveau 1

18.1 Énoncé

Déterminer si une année est bissextile.
Écrire une fonction qui prend une année (un entier) en argument et retourne True si elle est bissextile, False sinon. Attention une année n'est pas bissextile tout les 4 ans, c'est plus délicat.

18.2 Entrée

L'entrée ne contient qu'un seul entier : l'année>0.

18.3 Sortie

True si l'année est bissextile, False sinon.

18.4 Exemples d'entrée/sortie

```
# Dans la console Python
>>>  is_bissextile(2000)
True
>>>  is_bissextile(2042)
False
```

18.5 Rendu

— Un code python dans le bon format.
— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

19 Hauteur de jetons – Niveau 2

19.1 Énoncé

Hauteur des jetons dans une grille de puissance 4.
On donne une grille de Puissance 4 : un tableau de taille N par M, de 0 et de 1, où les 1 sont des jetons, de couleur indifférenciée, et les 0 les trous ; vous devez trouver la hauteur maximale atteinte par les jetons.

19.2 Entrée

Une grille représentée par un tableau (NxM) avec une liste de liste de 0 et de 1 : 0 ou 1, représentant respectivement les jetons et trous de la grille.

19.3 Sortie

La sortie contient un entier : la hauteur maximale atteinte par les jetons.

19.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
grille=[
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0],
[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
]
```

```
# Dans la console Python
>>>  hauteurJeton(grille)
2
```

19.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

20 Sudoku – Niveau 3

20.1 Énoncé

Déterminer si une grille de sudoku est valide.
Le Sudoku est un jeu qui est devenu très populaire récemment. Il se présente sous la forme d'un tableau de trois grilles par trois, elles mêmes composées de trois cases par trois. Le but du jeu consiste à remplir les cases de chiffres allant de 1 à 9 sans qu'un même chiffre apparaisse plus d'une fois par ligne, colonne, et grille.
On vous donne donc un tableau de taille 9 par 9, rempli d'entiers allant de 1 à 9. Vous devez écrire une fonction qui retourne True si ce tableau est un jeu de Sudoku correctement rempli (et False sinon).

20.2 Entrée

En entrée on vous donne une liste de liste de neuf lignes de neuf entiers.

20.3 Sortie

La sortie contient un booléen : False ou True.

20.4 Exemples d'entrée/sortie

```
#dans l'éditeur PYTHON
grilleSudoku=[
[1, 2, 3, 4, 5, 6, 7, 8, 9],
[4, 5, 6, 7, 8, 9, 1, 2, 3],
[7, 8, 9, 1, 2, 3, 4, 5, 6],
[2, 3, 4, 5, 6, 7, 8, 9, 1],
[5, 6, 7, 8, 9, 1, 2, 3, 4],
[8, 9, 1, 2, 3, 4, 5, 6, 7],
[3, 4, 5, 6, 7, 8, 9, 1, 2],
```

```
        [6, 7, 8, 9, 1, 2, 3, 4, 5],
        [9, 1, 2, 3, 4, 5, 6, 7, 8]]
```

```
# Dans la console Python
>>>  is_sudoku(grilleSudoku)
True
```

20.5 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.

L'héritière informaticienne – Qualification 2007
Niveau 4 Énoncé
Le problème de Josephus.
Un vieil homme qui a beaucoup (vraiment beaucoup) de descendants (N) veut choisir lequel sera son héritier. Il les dispose en cercle, les numérote de 0 à N-1, et se met à en éliminer un sur K jusqu'à ce qu'il n'en reste qu'un... À quelle position doit se placer l'informaticienne de la famille pour être celle qui est choisie?
Si le vieil homme a sept enfants et qu'il en élimine un sur trois, il compte 0, 1, élimine le 2, compte 3, 4, élimine le 5, compte 6, 0, élimine le 1, compte 3, 4, élimine le 6 (2 et 5 déjà éliminés), et ainsi de suite (il élimine 4 et 0). La fille chanceuse (ou informaticienne) se place donc en numéro 3.
De même, la fonction renvoie 0 pour N=1 (c'est le seul restant), 1 pour N=2 et K=3 (on compte 0, 1, 0, donc 0 éliminé), 2 pour N=5 et K=2, et 37 pour N=42 et K=7. Contraintes
0 < N 0 < K
Entrée
La première ligne de l'entrée contient les deux entiers N et K. Sortie
La sortie contient un entier : la position de l'héritière. Contraintes d'exécution
Utilisation mémoire maximum 500 kilo-octets Temps d'exécution maximum 1250 millisecondes
Exemples d'entrée/sortie
Exemple d'entrée
7 3
Exemple de sortie
3

20.6 Rendu

— Un code python dans le bon format.

— Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

— Un algorithme en pseudo-code écrit en latex qui résume l'idée de votre code.