Conseils :

- Prenez le temps de bien faire les choses et **rendez une copie propre**. Des points seront mis pour le soin (environ 1,5 point sur 20).
- Aucun ordre n'est imposé pour la résolution, par contre, rendez les problèmes dans l'ordre. Numérotez avec rigueur les questions que vous traitez.
- L'usage des **calculatrices est interdit**.

Quelques exercices

Q1. Écrire une fonction **positif_all** *prenant en argument* ***une liste de nombres non vide*** *qui renvoie* ***True*** *si tous les nombres sont positifs et* ***False*** *sinon.*

Q2. Ecrire une fonction **cherche_lettre** *prenant en argument* ***une chaine de caractères*** **chaine** *et un* ***caractère*** **x** *qui renvoie sous forme d'une* ***liste*** *le ou les indices du caractère x dans la chaine de caractères. La fonction renverra une liste vide si le caractère n'est pas dans la chaine de caractères.* Par exemples cherche_lettre (« informatique », « i ») doit renvoyer [0,8], cherche_lettre (« informatique », « a ») doit renvoyer [ ].

Q3. Soit la suite définie par $u_0 = 2$ *et* $u_{n+1} = 1 + 2u_n^2$ *. Définir une fonction* **seuil** *qui prend en argument* ***un nombre k*** *et qui renvois* ***le plus petit entier m*** *tel que* $u_m \geq k$*. Justifier précisément pourquoi votre programme renvoie le bon résultat.*

Q4. On dispose d'une liste dans laquelle on souhaite supprimer certains éléments. Pour l'exemple, on prendra L = [1 ,0 ,1 ,2 ,0 ,2 ,1] dont on cherche à supprimer les 2. On devra donc obtenir après modification L = [1 ,0 ,1 , 0, 1]. Définir une fonction **nettoieListe** *qui prend en argument* ***une liste de nombre et un nombre*** *et qui renvoie* ***la même liste*** *sans le nombre dans la liste. Nettoyer la liste avec une boucle while, utiliser la fonction « del », qui s'écrit :* del L[i] pour supprimer l'élement d'indice i dans le liste L.

Q5. Ecrire un script python (pas besoin forcément d'une fonction) qui crée un ***dictionnaire fr_en*** *qui contient en clé les mots français* ***informatique, je, aime*** *et en valeur les mots anglais* ***computing, I, love****.*

Q6. *Ecrire une fonction* **traduction** *qui prend en* ***argument une « phrase » sous forme de liste*** *de mots et* ***un dictionnaire*** *qui contient les traductions mot à mot ; votre fonction renverra la traduction mot à mot de* ***la phrase sous frome de liste****.* Par exemple traduction ([« je » , « aime » , « informatique] , fr_en) doit renvoyer la liste [« I » , « love », « computing »].

Q7. Ecrire une fonction ***inverse_dico*** *qui prend en un argument* ***un dictionnaire*** *de traduction mot à mot (fr_en par exemple) et qui renvoie le* ***dictionnaire*** *pouvant réaliser la traduction de la deuxième à la première langue (en_fr par exemple). On supposera qu'il n'aura pas de doublon dans les valeurs.*

Q8. Ecrire une fonction ***inventaire*** *qui prend en argument une liste d'éléments qui peuvent être présents plusieurs fois ([« a », « b », « a », « c », « a », « c »]) et qui renvoie un* ***dictionnaire*** *qui aura pour clé les éléments de la liste sans doublon et pour valeurs le nombre de chaque éléments.* Par exemple inventaire ([« a », « b », « a », « c », « a », « c »]) doit renvoyer { « a » : 3 , « b » : 1 , « c » : 2 }

10 km de la Saint Nicolas

Lors de la course de la Saint-Nicolas ayant lieu à Nancy, chaque coureur reçoit lors de son inscription un numéro de dossard unique ainsi que le dossard correspondant. Celui-ci est muni d'une puce électronique. Cette puce est initialisée sur la ligne de départ, et permet d'enregistrer automatiquement le temps à l'arrivée lors du passage du coureur sur le tapis de chronométrage présent sur la ligne d'arrivée.

Partie A : Gestion de la course et statistiques

Les données collectées par la puce sont rassemblées dans des listes contenant n éléments, n étant le nombre de coureurs. Une liste **dossard** contient les numéros de dossard des coureurs qui ont franchi la ligne d'arrivée. Une liste **temps** contient les temps réalisés par ces différents coureurs, exprimés en secondes. L'indice d'un concurrent sera par définition l'indice qui permet d'accéder à son dossard dans la liste **dossard** et à son temps dans la liste **temps**.

Exemple 1 :

indice	0	1	2	3	4
dossard	12	23	66	78	85
temps	8300	7256	3010	4505	6701

Dans l'exemple 1, le coureur d'indice 1 porte le dossard numéro 23 et a terminé la course en 7256 s.

Q9. Écrire une fonction **vainqueur** *qui prend en argument* ***les deux listes, dossard et temps****, et qui renvoie le* ***dossard*** *du vainqueur de la course.*

Q10. Écrire une fonction **temps_moyen** *qui prend en argument la liste* ***temps*** *qui renvoie la* ***valeur moyenne des temps*** *mis par les coureurs pour franchir la ligne d'arrivée.*

Gaston, étudiant en école d'ingénieur et bénévole pour l'organisation de la course, pense qu'il serait plus simple d'utiliser un dictionnaire pour gérer les numéros de dossard et les temps correspondant. Les clés correspondent aux numéros de dossard et les valeurs aux temps.
Il crée donc le dictionnaire suivant à partir de l'exemple 1:
course = {12 : 8300, 23 : 7256, 66 : 3010, 78 : 4505, 85 : 6701}

Partie B : Alerte météo

Lors d'une édition de la course de la Saint-Nicolas, suite à un violent orage le système d'acquisition des temps d'arrivée via les puces électroniques tomba en panne. L'organisation fut obligée de mettre les coureurs à l'abri. Ceux- ci se trouvaient alors encore dans le couloir d'arrivée, dans l'ordre de passage de la ligne, mais sans que l'on ait eu le temps de noter leurs positions. Le speaker eut juste le temps de demander à chacun des concurrents de retenir le dossard de celui qui se trouvait immédiatement devant lui (et qui était donc arrivé juste avant lui). Une fois à l'abri, on a demandé à chaque concurrent de remplir un formulaire. Chacun devait ainsi donner son dossard et celui du concurrent arrivé juste avant lui (si celui- ci existe).

Les données extraites des formulaires ont permis de construire deux listes de n éléments :

- la liste **dossard** qui contient le numéro de dossard de chaque concurrent.
- la liste **preced** qui contient le numéro de dossard du concurrent arrivé immédiatement avant lui. Pour le gagnant de la course, la valeur de preced est None.

Le but de cette partie est d'arriver à reconstituer le classement de la course en utilisant les tableaux **dossard** et **preced**.
Dans cette partie, on suppose qu'il n'y a aucune erreur (ni aucune triche !) dans le remplissage du formulaire.
L'indice d'un concurrent sera, par définition l'indice qui permet d'accéder à son dossard dans le tableau dossard.

Exemple 2 :

indice	0	1	2	3	4
dossard	12	23	66	78	85
preced	23	85	None	66	78

Dans l'exemple 2, le coureur d'indice 1 porte le dossard numéro 23 et est arrivé juste après le coureur de dossard 85.

Q11. Quel est le dossard du coureur ayant gagné la course ?
Quel est le dossard du coureur ayant terminé deuxième ?
Reconstituer le classement de la course du 1er au dernier coureur.

Q12. *Écrire une fonction* **verifie** *prenant en argument* ***la liste*** *dossard et* ***un entier*** *numero, qui renvoie* ***True*** *si le numero correspond bien à un dossard présent dans la course et* ***False*** *sinon.* Dans l'exemple 2, verifie(dossard,78) renvoie True.

Q13. Écrire une fonction **cherche** *qui prend en argument* ***une liste L et un entier x****, qui renvoie* ***l'indice de l'élément x*** *dans la liste L, si ce dernier est présent ; None sinon.*

Dans l'exemple 2, cherche(dossard,66) doit renvoyer 2 tandis que cherche(dossard,45) doit renvoyer None.

Q14. Écrire une fonction **gagnant** *prenant en argument* ***les listes dossard et preced*** *et qui renvoie* ***le dossard du vainqueur*** *de la course.*

Q15. Écrire une fonction **perdant** *prenant en argument* ***les listes dossard et preced****, et qui renvoie* ***l'indice et le dossard*** *du concurrent arrivé le dernier, sous forme de tuple.*

Pour obtenir le classement de la course, on va construire un tableau **ind_prec** : pour chaque indice i, **ind_prec[i]** correspond à l'indice du concurrent arrivé immédiatement avant le concurrent d'indice i. Les valeurs stockées dans le tableau **ind_prec** sont donc des indices. Pour le coureur arrivé en premier, on attribuera None dans le tableau **ind_prec**.

Q16. Dans l'exemple 2, donner le tableau ind_prec qui correspond aux listes dossard et preced.

Q17. On revient au cas général. Écrire une fonction **construit_ind_prec** *prenant en argument* ***les listes dossard et preced*** *et qui renvoie* ***le tableau ind_prec****.*

Q18. Écrire une fonction **classement** *prenant en argument* ***les listes dossard et preced*** *et qui renvoie* ***le classement de la course*** *sous la forme d'une liste constituée des numéros de dossard dans l'ordre inverse d'arrivée.* Dans l'exemple 2, classement(dossard,preced) doit renvoyer la liste [12, 23, 85, 78, 66].

Gaston qui veut aider à établir le classement, fournit le script ci-dessous.

```
def fonction(x,y):
    s=0
    t=0
    for i in x:
        if i != None:
            s+=i
    for i in y:
        if i != None:
            t+=i
    return s-t
```

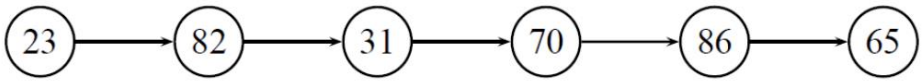
Q19. De quels types doivent être les arguments x et y ?

Q20. Appliquer cette fonction à l'exemple 2, c'est dire appliquer fonction (dossard, preced) et donner la valeur renvoyée. *Quel est le rôle de cette fonction ?*

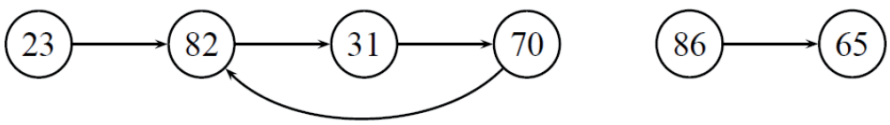
Partie C : Coureur peu lucide ou quelque peu tricheur

On suppose maintenant qu'un concurrent s'est trompé en indiquant le dossard de celui qui est arrivé devant lui. Il a donc donné un mauvais numéro, mais celui-ci est néanmoins existant. Si le vainqueur se trompe c'est qu'il a donné un numéro au lieu de ne rie, donner.

Exemple 3 :
Sans cette erreur, on peut représenter par le schéma suivant le classement dans l'ordre inverse d'arrivée :



On suppose maintenant que le concurrent de dossard 70 a commis une erreur, il a indiqué avoir vu le dossard 82 devant lui. Le schéma devient alors :



Q21. Écrire une fonction **erreur_vainqueur** *qui prend en argument* ***la liste preced*** *et qui renvoie* ***True*** *si le vainqueur de la course s'est trompé,* ***False*** *sinon.*

Q22. Écrire une fonction **cherche_doublon** *prenant en argument* ***la liste preced*** *et qui permet de repérer si cette liste contient deux valeurs identiques. La fonction renverra la valeur si c'est le cas, None sinon.*

On suppose désormais disposer d'une fonction **dossard_devant(dossard, preced,d)** qui prend en argument les listes **dossard** et **preced** ainsi qu'un dossard **d**. Cette fonction renvoie le dossard que le concurrent de dossard d a «vu » devant lui. Dans l'exemple précédent, **dossard_devant(dossard, preced, 86)** retourne 65.
On se place dans la situation où deux personnes différentes ont « vu » devant elles le dossard d'un même concurrent. On dispose donc du dossard de ce concurrent qui est stocké dans la variable doublon.
On suppose que le fautif a désigné le dossard d'un concurrent arrivé derrière lui (exemple 3).

Q23. Écrire une fonction **fautif***(dossard, preced, doublon) retournant* ***le dossard du fautif***