Composants fournis

Dans tout le sujet, on utilisera les variables globales initialisées de la manière suivante

```
objectif ← [[1,2,3],
            [4,5,6],
            [7,8,0]]

tab ← [[2, 6, 3],
       [7, 0, 5],
       [4, 1, 8]]

ligne_case_vide ← 1

colonne_case_vide ← 1
```

La variable `objectif` ne sera pas modifiée : elle représente l'état du jeu lorsque la partie est terminée.
La variable `tab` est un tableau d'entiers représentant le jeu.
La cellule contenant la valeur 0 représente la case vide.
L'indice de ligne et l'indice de colonne de cette case vide sont recopiées dans les deux variables `ligne_case_vide` et `colonne_case_vide`

On donne l'algorithme d'une procédure `bas()` qui modifie le contenu des variables globales `tab` et `ligne_case_vide` ci-dessus

```
procédure bas()
variables locales : i et j entiers
début
    i ← ligne_case_vide
    j ← colonne_case_vide
    tab[i][j] ← tab[i+1][j]
    tab[i+1][j] ← 0
    ligne_case_vide ← i+1
fin
```

Partie A

Cette partie est à traiter sans machine. Les réponses sont à écrire sur la feuille réponse jointe.

Question A1
On exécute une première fois l'instruction : `bas()`

a) Déterminer le contenu des variables globales `tab`, `ligne_case_vide` et `colonne_case_vide` à la fin de l'exécution de l'instruction `bas()`.
=> répondre sur la feuille réponse.

b) quelle valeur du tableau s'est finalement « déplacée vers le bas » ?
=> répondre sur la feuille réponse.

Question A2
On exécute une deuxième fois l'instruction : `bas()`

Expliquer pourquoi cette instruction provoquera une erreur
=> répondre sur la feuille réponse.

Question A3
Proposer une modification de l'algorithme de la procédure bas() qui évite l'erreur de la question 2 : si le déplacement vers le bas est impossible, alors la procédure ne changera le contenu d'aucune des variables.
=> répondre sur la feuille réponse.

Question A4
Ecrire l'algorithme d'une procédure `haut()` qui :

- effectue, quand c'est possible, le déplacement vers le haut de la valeur 0 du tableau `tab`.
- met à jour les variables globales `ligne_case_vide` et `colonne_case_vide`

=> répondre sur la feuille réponse.

Partie B

Dans cette partie, vous devez utiliser un ordinateur.
Le fichier `TPTaquin.py` vous est fourni : vous devez travailler dans ce fichier. Pensez à enregistrer régulièrement votre travail.

Avant de commencer cette partie, vous pouvez exécuter le fichier fourni est tester les commandes

```
>>> affiche()
>>> bas()
>>> affiche()
```

Question B1 : les quatre coups possibles

a) Dans le code en python, modifier la procédure `bas()` en tenant compte de la question A3

b) En suivant votre algorithme de la question A4, écrire en python une procédure `haut()`

c) Écrire de même deux procédures `gauche()` et `droite()` qui :

- effectuent, quand c'est possible, le déplacement vers la gauche (ou droite) de la valeur 0 du tableau `tab`.
- mettent à jour les variables globales `ligne_case_vide` et `colonne_case_vide`

Penser à tester votre programme

Question B2 : contrôle de jeu

a) Compléter la procédure `gameover()` pour qu'elle renvoie True si la variable tab a le même contenu que la variable objectif (et donc que la partie est terminée).

b) Compléter la procédure `melange()` qui doit effecuter le « mélange » du jeu au hasard : cette procédure sera appelée au début de chaque nouvelle partie

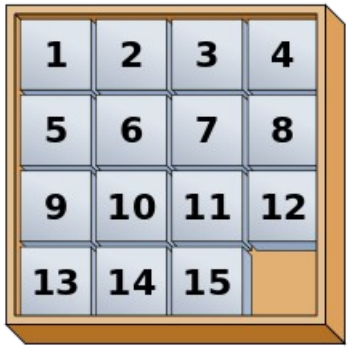
- choisir au hasard un certain nombre de « coups » : gauche() droite() bas() ou haut()
- rappel : la fonction `randint(a,b)` renvoie un entier aléatoire compris entre `a` et `b` (inclus).

Lorsque tout ceci est fait, vous pouvez lancer : `ExecuteInterface.py`

Sujet : le jeu du Taquin

Le jeu du taquin est composé de pièces numérotées, et disposées dans un cadre en laissant un emplacement vide.

Le but du jeu est de ranger les pièces dans l'ordre croissant en un nombre fini de coups. Chaque coup consiste à mettre à la place vide une des pièces voisines de cette case.



Dans la suite, on considère un jeu formé de 8 pièces numérotées de 1 à 8 et disposées dans un carré (3×3).

Feuille réponse de la partie A

Question A1

On exécute une première fois l'instruction : `bas()`

a) Déterminer le contenu des variables globales à la fin de l'exécution de l'instruction `bas()`.

```
Tab = [[   ,   ,   ],
       [   ,   ,   ],
       [   ,   ,   ],]

ligne_case_vide = ...

colonne_case_vide = ...
```

b) quelle valeur du tableau s'est finalement « déplacée vers le bas » ?

Question A2

On exécute <u>une deuxième fois</u> l'instruction : `bas()`

Expliquer pourquoi cette instruction provoquera une erreur

Question A3

Proposer une modification de l'algorithme de la procédure bas() qui évite l'erreur de la question 2 : si le déplacement vers le bas est impossible, alors la procédure ne changera le contenu d'aucune des variables.

```
procédure bas()
variables locales : i et j entiers
début

    i ← ligne_case_vide
    j ← colonne_case_vide
    tab[i][j] ← tab[i+1][j]
    tab[i+1][j] ← 0
    ligne_case_vide ← i+1
fin
```

Question A4

Ecrire l'algorithme d'une procédure `haut()` qui :

- effectue, quand c'est possible, le déplacement vers le haut de la valeur 0 du tableau `tab`.
- met à jour les variables globales `ligne_case_vide` et `colonne_case_vide`