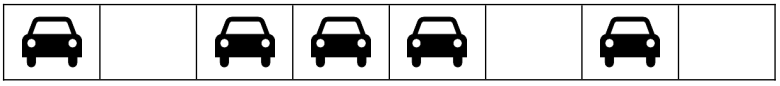
NOMBRE DE PLACES LIBRES DANS UN PARKING

L'objectif de cette épreuve est d'écrire un programme permettant de calculer et d'afficher le nombre de places libres dans un parking de 4 allées de 8 places.
Chaque allée est associée à une variable de type tableau d'entiers de longueur 8.
Chaque place libre est représentée par la valeur 1,
et chaque place occupée est représentée par la valeur 0.
Par exemple, le tableau d'entiers : [0, 1, 0, 0, 0, 1, 0, 1] représente l'allée suivante :



Dans tout le sujet, on considère :

- la variable `lane`, de type tableau d'entier de longueur 8 :

  `lane` ← [0, 1, 0, 0, 0, 1, 0, 1]

  on rappelle que pour un tableau, la numérotation des indices commence à 0.
  Par exemple, `lane[5]` correspond au nombre 1.

- la variable parking, de type tableau à deux dimensions, de 4 lignes et 8 colonnes correspondant à l'occupation d'un parking.

```
parking ← [[0, 0, 0, 0, 0, 0, 0, 0],
           [0, 1, 0, 0, 0, 1, 0, 0],
           [0, 1, 1, 0, 0, 1, 1, 1],
           [1, 1, 1, 1, 1, 1, 1, 1]]
```

  Ainsi, `parking[1][5]` correspond au nombre **1** en gras ci-dessus, et signale une place libre dans l'allée d'indice 1, à l'emplacement d'indice 5.

Partie A (30 minutes)

Cette partie doit être traitée avant d'accéder à l'ordinateur

Toutes les réponses sont à écrire sur les feuilles-réponse pages 5 à 8

A.1 On souhaite déterminer s'il y a au moins une place libre dans une rangée du parking.

On considère la fonction `occup`, dont l'algorithme est donné ci-dessous :

- le paramètre est un tableau d'entiers de longueur 8
- la fonction `occup` doit renvoyer VRAI s'il y a au moins une place libre dans la rangée représentée par `tab`, et elle doit renvoyer FAUX sinon.

```
fonction occup (tab : de type tableau d'entiers de longueur 8)

variables
     i  de type entier
     val de type entier
     test de type booléen

début_de_fonction
     i ← 0
     TANT QUE i < 8 :
          val ← tab[ i ]
          SI val = 1 :
               test ← VRAI
          SINON :
               test ← FAUX
          fin_du_si
     fin_du_tant_que
     retourner test
fin_de_fonction
```

A.1.a Déterminer la valeur renvoyée par `occup([1, 0, 0, 0, 1, 1, 0, 0])`

- effectuer le suivi des variables, sur la feuille réponse page 5
- expliquer pourquoi la valeur retournée par la fonction n'est pas celle attendue.

A.1.b. Proposer en annexe page 5 une version corrigée de la fonction `occup`.

A.2.

On considère la fonction mystere dont le paramètre est un tableau d'entiers de longueur 8, et dont l'algorithme est donné ci-dessous :

```
fonction mystere (tab : de type tableau d'entiers de longueur 8)

variables
     resu, i, val   de type entier

début_de_fonction
     resu ← 0
     POUR i allant de 0 à 7 :
          val ← tab[ i ]
          resu ← resu + val
     fin_pour
     retourner resu
fin_de_fonction
```

En exécutant la fonction mystere avec le paramètre `[1, 0, 0, 0, 1, 1, 0, 0]`,

- compléter le tableau de suivi des variables sur l'annexe page 7
- déterminer la valeur renvoyée par la fonction
- interpréter cette valeur dans le contexte de l'exercice

A.3.

On considère les quatre rangées du parking.

En page 8 de l'annexe, écrire l'algorithme (ou le code python) d'une fonction `places_dispo` :

- qui prend pour paramètre un tableau d'entiers de 4 lignes et 8 colonnes
- qui retourne le nombre de places libres dans chaque allée sous la forme d'un tableau de quatre entiers.

Par exemple, l'exécution de `places_dispo` avec le paramètre parking de la page 1 renvoie le tableau `[0,2,5,8]`

Partie B (30 minutes)

Cette partie doit être traitée sur ordinateur

B1. Programmer les fonctions définies dans la partie A

- fonction `occup_corrige` de la question **A.1.b**
- fonction `mystere`, question **A.2**
- fonction `places_dispo`, questions **A.3**

B2. Tester le fonctionnement de ces fonctions, en utilisant les paramètres `lane` et `parking`.

FEUILLES RÉPONSE DE LA PARTIE A
à remettre à l'examinateur

NOM : .....

A.1.a.
Suivi des variables lors de l'exécution de : `occup([1, 0, 0, 0, 1, 1, 0, 0])`

valeur de `tab` :

i										
val										
test										

- Valeur renvoyée par la fonction :

  .....

- pourquoi ce n'est pas le résultat attendu :

  .....

  .....

A.1.b Proposer une version corrigée de la fonction occup : il est possible de rayer et/ou réécrire toute partie du code fourni.

```
fonction occup_corrige (tab : de type tableau d'entiers de
longueur 8)

variables
     i  de type entier
     val de type entier
     test de type booléen

début_de_fonction

     i ←  0

     TANT QUE i < 8 :

          val ← tab[ i ]

          SI val = 1 :

               test ← VRAI

          SINON :

               test ← FAUX

          fin_du_si

     fin_du_tant_que

     retourner test


fin_de_fonction
```

A.2

En exécutant la fonction mystere avec le paramètre `[1, 0, 0, 0, 1, 1, 0, 0]`,

- compléter le tableau de suivi des variables

valeur de tab :

`i`										
`val`										
`resu`										

- valeur renvoyée par la fonction : .....
- interprétation cette valeur dans le contexte de l'exercice : .....

.....

.....

A.3. Algorithme (ou code python) d'une fonction `places_dispo` :

- qui prend pour paramètre un tableau d'entiers de 4 lignes et 8 colonnes
- qui retourne le nombre de places libres dans chaque allée sous la forme d'un tableau de quatre entiers.

Par exemple, l'exécution de `places_dispo` avec le paramètre parking de la page 1 renvoie le tableau `[0,2,5,8]`