# AP1/M1102 – DS 1 (mardi 10 novembre 2015)

*Aucun document et aucune machine*

**N'oubliez pas d'indiquer vos noms/prénoms sur les copies. Une copie par exercice.**

**Lisez bien l'ensemble de l'énoncé avant de commencer.**

---

## Exercice 1 : Les pangrammes (16 pts)

Un pangramme est une phrase comportant au moins une fois chaque lettre de l'alphabet. Par exemple, la phrase "*Portez ce vieux whisky au juge blond qui fume*" est un pangramme.

```
void testEstPangramme() {
  assertTrue( estPangramme("portez ce vieux whisky au juge blond
qui fume"));
  assertTrue( estPangramme("monsieur jack vous dactylographiez
bien mieux que votre ami wolf"));
  assertTrue( estPangramme("buvez de ce whisky que le patron juge
fameux"));
}
```

Afin de vous aider à écrire ce programme, nous l'avons décomposé en différentes fonctions.

Q1 (1pt) : Écrivez une fonction `estLettre` qui teste si un caractère passé en paramètre est bien une lettre minuscule.

```
void testEstLettre() {
  assertFalse( estLettre('Z') );
  assertTrue  ( estLettre('a') );
  assertTrue  ( estLettre('v') );
  assertFalse( estLettre('!') );
}
```

Q2 (2pt) : Écrivez une fonction `positionLettre` qui renvoie soit l'indice du caractère cherché dans une chaîne si il est présent, soit -1.

```
void testPositionLettre() {
  assertEquals( -1, positionLettre('c', "azerty") );
  assertEquals(  3, positionLettre('r', "azerty") );
}
```

Q3 (2pt) : On cherche maintenant à supprimer un caractère à un indice donné dans une chaîne de caractères. Écrivez la fonction `supprimeLettre` permettant de retourner une chaîne privée du caractère dont l'indice est donné :

```
void testSupprimeLettre() {
  assertEquals( "onjour", supprimeLettre(0, "bonjour") );
  assertEquals( "onour",  supprimeLettre(2, "onjour") );
}
```

Q4 (3pt) : Ecrivez une fonction `genereAlphabet` qui renvoie une chaîne de caractère contenant la liste de toutes les lettres de l'alphabet.

```
void testGenereAlphabet() {
  assertEquals("abcdefghijklmnopqrstuvwxyz", genereAlphabet());
}
```

**Remarque**: `'a'` a pour code ASCII `97`, mais vous pouvez aussi traiter cette question sans disposer de cette information.

Q5 (4pts). A partir de l'ensemble de ces fonctions (vous pouvez les utilisez même si vous n'avez pas su les écrire), écrivez maintenant la fonction `estPangramme` qui teste si une chaîne est un pangramme ou non. Vous pouvez recopier le code que vous avez donné dans les questions précédentes ou simplement appeler les fonctions précédentes pour écrire `estPangramme`.

```
void testEstPangramme() {
  assertTrue( estPangramme("portez ce vieux whisky au juge blond
qui fume"));
  assertTrue( estPangramme("monsieur jack vous dactylographiez
bien mieux que votre ami wolf"));
  assertTrue( estPangramme("buvez de ce whisky que le patron juge
fameux"));
}
```

**Q6.** (4pts) Un pangramme est dit remarquable s'il contient une unique fois chacune des lettres de l'alphabet. Modifiez la fonction précédente pour écrire la fonction `estRemarquable` qui détermine si un pangramme est remarquable.

---

## Exercice 2 : Conversion de nombres (8 pts)

Nous souhaitons dans cet exercice procéder à des changements de base. Comme vous le savez, les nombres entiers sont représentés en base 2 à l'aide de deux symboles, 0 et 1. Pour cet exercice, vous considérerez que vous disposez des variables suivantes :

```
int nombre;          // pour représenter les nombres en base 10
String binaire = ""; // pour représenter les nombres en base 2
```

Q1 (1pt) : Expliquez comment vous déterminez la valeur en base 2 du nombre `9` (base 10).

Q2 (2pts) : Ecrivez l'algorithme qui effectue la conversion d'un nombre naturel (ie. positif ou nul) en base 10 vers sa représentation en base 2. Vous supposerez que la variable `nombre` est déjà

initialisée et vous veillerez à ce que le résultat de la conversion se trouve dans la variable `binaire`.

Q3 (1pt) : Donnez la valeur en base 10 du nombre binaire `110` (base 2).

Q4 (4pts) : Ecrivez l'algorithme qui effectue la conversion d'un nombre en base 2 en base 10. Vous supposerez que la variable `binaire` est déjà initialisée correctement (c'est à dire qu'elle ne contient que des `0` et des `1`) et vous veillerez à stocker le résultat de la conversion dans la variable `nombre`. **Vous ne devez pas utiliser la fonction `pow` pour cet exercice**.

---

## Exercice 3 : Quelques manipulations sur les tableaux (16 pts)

Dans cet exercice, nous allons manipuler des tableaux de caractères.

Q1(1pt) : Définissez un tableau à une dimension de caractères nommé `phrase`.

Q2 (1pt) : Allouez 17 cases au tableau `phrase`.

Q3 (2pts) : Donnez le code d'une boucle permettant de copier le contenu de la variable:

```
String unePhrase = "Les jours heureux";
```

dans le tableau `phrase` (que vous considérerez comme déjà alloué).

Q4 (2pts) : Définissez maintenant la fonction générale permettant de copier le contenu d'une chaîne de caractères dans un nouveau tableau de caractères. Voici la signature de cette fonction:

```
char[] copier(String phraseChaine)
```

Q5 (3pts) : Définissez une fonction `boolean equals(char[] mot1, char[] mot2)` permettant de tester l'égalité de deux tableaux de caractères. Cette fonction prend en paramètre les deux tableaux à comparer et retourne soit `true` si le contenu des deux tableaux est identique, soit `false` si cela n'est pas le cas.

Q6 (3pts) : Définissez une fonction `int[] indicesOccurences(char[] unePhrase, char unCaractere)` qui retourne l'indice de la première et de la dernière occurrence du caractère `unCaractere` dans un tableau d'entiers. Si la lettre n'est pas présente dans la phrase, le tableau retourné devra contenir les valeurs −1. Il est acceptable de parcourir l'ensemble du tableau pour cette première version.

Q7 (4pts) : Ré-écrivez la fonction de la question précédente en n'utilisant qu'une seule boucle et en la quittant dès que les deux occurrences sont trouvées.