UNIVERSITÉ PARIS DIDEROT - LICENCE 1 - IF1 - PARTIEL - 28 NOVEMBRE 2009 - 08^{30} 10^{30}

Aucun document. Aucune machine. Les six exercices sont indépendants.
Les questions de chaque exercice peuvent être traitées indépendamment.
Les morceaux de code Java devront être clairement présentés, indentés et commentés.
Les quatorze *méthodes* demandées sont des *méthodes statiques*, également appelées *fonctions*.

Exercice 1 Préciser ce qu'affiche le morceau de code suivant :

```
int m=3, n=2, p;
float f,g;
p=3*m*n; m=n+3; n=5;
Deug.println(p); Deug.println(m); Deug.println(n);
m=n/p; f=p/n;
Deug.println(m); Deug.println(f);
g=n; f=p/g;
Deug.println(f);
```

Exercice 2 Le virus H1N1 de la grippe A a subi une mutation importante et les vaccins disponibles ne sont plus adaptés. Aussi, pour une commande de 100 000 à 499 999 doses, un laboratoire concède un rabais de 3%. Si l'on achète plus de 500 000 doses, le rabais est de 5%.

Dans cet exercice, le choix des types Java est laissé au programmeur !

1. Écrire une méthode `arrondi` qui arrondit au centième le plus proche, c'est-à-dire prend un réel en argument et retourne le plus proche réel ayant au plus 2 chiffres après la virgule.
2. Écrire une méthode `facture` qui prend comme arguments le prix unitaire du vaccin (en euro) et le nombre de doses achetées et retourne le prix à payer (arrondi au cent d'euro).

Exercice 3 On s'intéresse aux *ordinaux anglais abrégés*, où le nombre (le cardinal) est écrit en chiffres :

1st, 2nd, 3rd, 4th, . . . , 9th, 10th, 11th, 12th, . . . , 19th, 20th, 21st, 22nd, 23rd, . . .

Pour déterminer le suffixe, on regarde le dernier chiffre du nombre : si c'est 1, on ajoute le suffixe "st" ; si c'est 2, le suffixe est "nd" ; si c'est 3, le suffixe est "rd" ; sinon le suffixe est "th".
Il y a cependant une exception : si l'avant-dernier chiffre du nombre est 1, le suffixe est toujours "th".

1. Écrire une méthode `afficherOrdinal` qui prend un entier de type `int` en argument et affiche l'ordinal anglais abrégé correspondant.
2. Écrire une méthode `main` qui, après avoir demandé un entier à l'utilisateur, utilise la méthode `afficherOrdinal` pour afficher, si l'entier est `3` par exemple, le message :

```
Here is the 1st line.
Here is the 2nd line.
Here is the 3rd line.
```

Exercice 4 On se propose de construire une méthode de tri, appelé *tri par sélection*.

1. Écrire une méthode `maximum` qui prend en arguments deux entiers `a` et `b` de type `int` et qui retourne le maximum des deux.
2. Écrire une méthode `maximum` qui prend en arguments un tableau `t` d'entiers de type `int` et deux indices `i` et `j` et qui, en supposant `i`$\leq$`j` valides et `t` non vide, retourne la plus grande valeur contenue dans `t` à un indice compris entre `i` et `j`.
3. Écrire une méthode `indice` qui prend en arguments un tableau `t` d'entiers de type `int` et une valeur `c` de type `int` et qui retourne le plus petit indice d'une occurrence de `c` dans `t` si une telle occurrence existe et retourne -1 sinon.

4. Écrire une méthode `echanger` qui prend en arguments un tableau `t` d'entiers de type `int` et deux indices `i` et `j` et qui échange dans `t` les éléments d'indice `i` et `j`.
5. Des trois méthodes précédentes, déduire une méthode `trier` qui prend en argument un tableau `t` d'entiers de type `int` et le trie : sélection de l'élément maximum et déplacement en dernière position, sélection de l'élément maximum sur la partie du tableau restant à trier et déplacement en avant dernière position, etc.

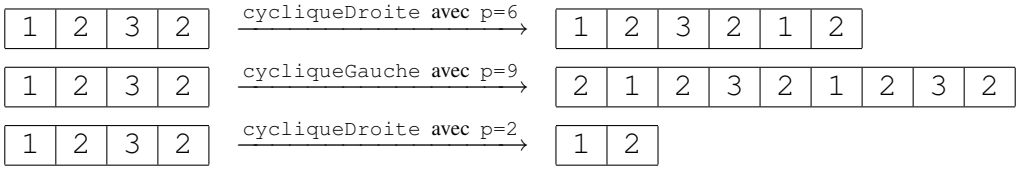
Exercice 5 On considère le jeu du *bizarre appartement vermillon*.

```
> javac BizarreAppartementVermillon.java
> java BizarreAppartementVermillon
Proposez une chose pouvant se trouver dans le bizarre appartement vermillon :
Crocodile
Non. Malheureusement. Une autre chose :
Gazelle
Effectivement. Une autre chose :
Pomme
Effectivement. Une autre chose :
PomME
Non. Malheureusement. Une autre chose :
Porte-manteau
Ouch ! Le mot est invalide. Au revoir.
>
```

On rappelle que, pour toute chaîne de caractères u et pour $0 \leq$ `i` $\leq$ `u.length()-1`, l'expression `Deug.charAt(u,i)` est de type `char` et a pour valeur le caractère d'indice `i` de `u`.

1. Écrire une méthode `estUnMotValide` qui prend en argument une chaîne de caractères `u` et teste si `u` ne contient que des caractères alphabétiques : `a`, `A`, `b`, `B`, ..., `z`, `Z`.
2. Écrire une méthode `contientUneLettreDoublee` qui prend en argument une chaîne de caractères `u` et teste si `u` contient une lettre doublée (deux caractères successifs identiques).
3. Écrire une méthode `main` qui invite l'utilisateur à proposer une chose pouvant se trouver dans le *bizarre appartement vermillon*, affiche la réponse et répète ceci tant que l'utilisateur entre un mot valide.

Exercice 6 À partir d'un tableau d'entiers et d'un entier `p` > 0, on construit le tableau de longueur `p` obtenu en répétant *cyclique*ment les éléments du tableau initial soit vers la *droite* soit vers la *gauche*.



1. Écrire la méthode `cycliqueDroite` associée au type tableau d'entiers de type `int`.
2. Écrire la méthode `cycliqueGauche` associée au type tableau d'entiers de type `int`.
3. Cette construction peut être définie pour d'autres types (type `String`, types entiers, autres types de tableaux, par exemple). En supposant que les méthodes associées portent toujours les noms `cycliqueDroite` et `cycliqueGauche`, décrire en justifiant l'exécution du morceau de code suivant :

```
String c = "bracadd";
long n = 2013144L;
Deug.println(cycliqueDroite(
        cycliqueGauche(cycliqueDroite(c,5),6)+cycliqueDroite(cycliqueGauche(c,9),1),
  (int)(cycliqueGauche(cycliqueDroite(n,7),4)%cycliqueDroite(cycliqueGauche(n,5),2))));
```