Aucun document. Aucune machine. Les six exercices sont indépendants. Le barème est indicatif.
Les morceaux de code Java devront être clairement présentés, indentés et commentés.
Les vingt-deux *méthodes* demandées sont des *méthodes statiques*, également appelées *fonctions*.
Les méthodes des classes `Deug` ou `System` sont interdites (sauf question 4 de l'exercice 5).

Exercice 1 (2 points) Expliquer ce que produit l'exécution du programme suivant :

```
import fr.jussieu.script.Deug;
class Kezako{
    static int f(int i){
        Deug.println("f prends " + i);
        return i + g(i*2);
    }
    static int g(int j){
        Deug.println("g prends " + j);
        return j-1;
    }
    public static void main(String[] args){
        int a = f(12); Deug.println("résultat " + a);
        a = f(g(f(3))); Deug.println("résultat " + a);
    }
}
```

Exercice 2 (3 points) On s'intéresse aux *fonctions booléennes ternaires*.

1. Donner le nombre total de telles fonctions. Justifier.
2. Écrire des méthodes `nxor1` et `nxor2` (indépendantes l'une de l'autre) chacune prenant trois arguments de type booléen et renvoyant `true` si et seulement si les trois arguments portent la même valeur de vérité.
   (a) le corps de `nxor1` ne doit contenir qu'une seule instruction `return` ;
   (b) le corps de `nxor2` ne doit utiliser aucun opérateur booléen (pas même l'opérateur ==).

Exercice 3 (4 points) Si `t` est un tableau d'entiers de longueur $\ell > 1$, le *tableau des écarts* de `t` est le tableau d'entiers de longueur $\ell - 1$ dont l'élément d'indice `i` est `t[i]-t[i+1]`.

Par exemple, le tableau des écarts de | 3 | 4 | 4 | 3 | 7 | est | −1 | 0 | 1 | −4 | .

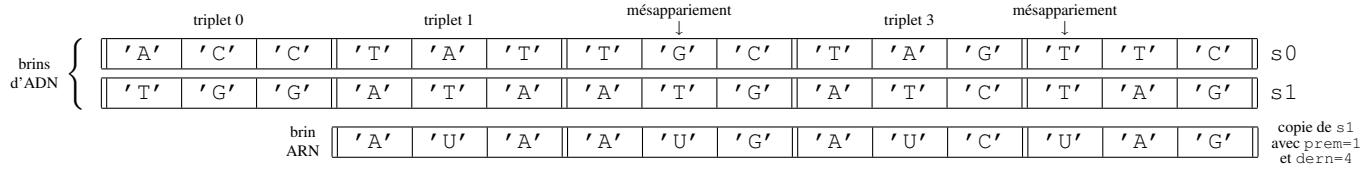
1. Écrire une méthode `ecarts` qui prend comme paramètre un tableau d'entiers dont la longueur est supposée strictement supérieure à 1 (il n'est donc pas nécessaire de le vérifier) et qui renvoie son tableau des écarts.

En itérant la construction des tableaux des écarts, à partir d'un tableau d'entiers `t` de longueur strictement supérieure à 1, on trouve un tableau de longueur 1 : l'unique élément de ce tableau est appelé *poids* de `t`.

Par exemple, le poids de | 3 | 4 | 4 | 3 | 7 | est 6.

2. Écrire une méthode `poids` qui prend comme paramètre un tableau d'entiers dont la longueur est supposée strictement supérieure à 1 et qui renvoie son poids.

Exercice 4 (5 points) L'ADN s'organise en deux *brins* parallèles constitués de *triplets* des quatre *bases* `A`, `T`, `C` et `G`. Les bases `A` et `T` (respectivement, les bases `C` et `G`) forment des *paires complémentaires* : les bases d'une paire se font face. L'ARN est lui constitué d'un seul brin obtenu comme copie partielle d'un brin ADN mais où la base `U` remplace la base `T`. De tels brins peuvent être représentés par des tableaux à une dimension de caractères :



1. Écrire une méthode `nbMesappariements` comptant le nombre de bases mal appariées entre deux brins ADN donnés `b0` et `b1` (et renvoyant `-1` si `b0` et `b1` ne sont pas de même longueur).
2. Écrire une méthode `transcription` qui renvoie le brin ARN construit à partir d'un brin ADN donné `b` et des indices `prem` et `dern` des premier et dernier triplets copiés.
3. La partie codante d'un brin ARN est strictement comprise entre un triplet de début (`AUG`) et un triplet de fin (`UAA`, `UAG` ou `UGA`), triplets les plus à gauche. Écrire une méthode `partieCodante` qui extrait la partie codante d'un brin ARN donné `b` si c'est possible et renvoie `null` sinon.

4a. Écrire une méthode `nbTryptophane` comptant le nombre de triplets `UGG` codant l'acide aminé tryptophane dans la partie codante d'un brin ARN donné `b`.

4b. L'acide aminé isoleucine est lui codé par `AUU`, `AUC` ou `AUA`. Écrire une méthode `nbIsoleucine`.

5. Écrire une méthode `satisfaitChargaff` testant si un brin ADN donné `b` satisfait les règles de Chargaff stipulant que les rapports entre les bases `A` et `T` d'une part et entre les bases `C` et `G` d'autre part sont proches de 1. Une marge d'erreur `epsilon` sera passée en argument.

Exercice 5 (7 points) Sur le principe du pousse-pousse, deux joueurs (`o` et `x`) enfilent (par le haut) à tour de rôle des jetons (`o` et `x`) dans une grille carrée jusqu'à obtenir strictement plus d'alignement(s) (suivant les lignes, les colonnes et/ou les deux diagonales). Les jetons poussés hors de la grille (par le bas) sont remis en jeu. Dans l'exemple de partie ci-contre, le joueur `o` débute et gagne la partie.
La grille est codée par un tableau carré contenant les entiers `-1` (désignant un jeton `o`), `0` (désignant une case vide) et `+1` (désignant un jeton `x`).

```
Entrer la taille : 3
    . . .
    . . .
    . . .
Joueur o, quelle colonne ? 1
    . o .
    . . .
    . . .
Joueur x, quelle colonne ? 1
    . x .
    . o .
    . . .
Joueur o, quelle colonne ? 0
    o x .
    . o .
    . . .
Joueur x, quelle colonne ? 2
    o x x
    . o .
    . . .
Joueur o, quelle colonne ? 1
    o o x
    . x .
    . o .
Joueur x, quelle colonne ? 1
    o x x
    . o .
    . x .
Joueur o, quelle colonne ? 2
    o x o
    . o x
    . x .
Joueur x, quelle colonne ? 2
    o x x
    . o o
    . x x
Joueur o, quelle colonne ? 2
    o x o
    . o x
    . x o
Le joueur o gagne!
```

1a. Écrire une méthode `jeton` qui prend en argument un entier `k` et renvoie le caractère `'o'` si `k` est strictement négatif, renvoie le caractère `'x'` s'il est strictement positif et renvoie `'.'` sinon.

1b. Écrire une méthode `toString` qui prend en argument un tableau carré de `-1`, `0`, `+1` et renvoie une chaîne de caractères représentant la grille associée.

2. Écrire une méthode `jouer` qui prend en arguments un joueur (`-1` ou `+1`), un indice de colonne et un tableau carré et met à jour ce dernier en fonction.

3a. Écrire une méthode `evaluation` qui prend en argument un tableau à une dimension de `-1`, `0`, `+1` et renvoie `-1` s'il s'agit d'un alignement de `-1`, renvoie `+1` s'il s'agit d'un alignement de `+1` et renvoie `0` sinon.

3b. Écrire des méthodes `diagonale`, `antidiagonale` et `colonne` qui prennent en argument un tableau carré (plus un indice `j` pour `colonne`) et renvoient le tableau correspondant respectivement à la diagonale (NO-SE), à l'antidiagonale (NE-SO) et à la colonne d'indice `j`.

3c. Écrire une méthode `evaluation` qui prend en argument un tableau carré de `-1`, `0`, `+1` et renvoie le nombre d'alignement(s) de `+1` moins le nombre d'alignement(s) de `-1`.

4. Écrire une méthode principale qui demande la taille de la grille, puis demande alternativement à deux joueurs quelle colonne jouer en affichant à chaque fois la grille obtenue, et annonce le vainqueur dès que possible.

Exercice 6 (6 points) On considère un tableau à 3 dimensions stockant les scores d'un championnat de handball. Pour `n` équipes, le tableau aura `n` lignes et `n` colonnes et dans chacune de ses cases on trouvera un tableau à une dimension de longueur 2 contenant le score d'un match (on ne tient pas compte de ce qui est stocké dans la diagonale). Ainsi, pour le tableau championnat `ch`, on trouvera dans `ch[i][j]` le score du match de l'équipe `i+1` contre l'équipe `j+1` et dans `ch[j][i]` le score du match de l'équipe `j+1` contre l'équipe `i+1`. De même, pour un score stocké, le premier entier de `ch[i][j]` sera le nombre de but(s) marqué(s) par l'équipe `i+1` dans le match l'opposant à l'équipe `j+1`. Finalement, on suppose que lorsqu'une équipe gagne un match, elle obtient 3 points, 1 seul point pour un match nul et 0 point dans le cas où elle perd le match.

1. Écrire une méthode `nombrePoints` qui prend en arguments un tableau championnat `ch` de côté `n` et le numéro d'une équipe (entre `1` à `n`) et qui renvoie le nombre de point(s) obtenu(s) par cette équipe pendant le championnat.

2. Écrire une méthode `stockerScore` qui prend en arguments un tableau championnat `ch` de côté `n`, le numéro `i` d'une équipe, le numéro `j` d'une autre équipe et le score du match de `i` contre `j` et qui met à jour le tableau `ch`.

3. Écrire une méthode `champion` qui prend en argument un tableau championnat `ch` et qui renvoie le numéro de l'équipe championne. Une équipe est championne si elle a strictement plus de points que toutes les autres. Si plusieurs équipes ont le même nombre de points, alors une équipe est meilleure si elle a marqué strictement plus de buts. Dans le cas d'égalité parfaite (même nombre maximum de points et même nombre maximum de buts marqués), la méthode renverra `0` pour signaler l'impossibilité de désigner un champion.