R1.01 - Initiation au développement
BUT Informatique
Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr
2023–2024

DS1 : Notions fondamentales d'algorithmique et programmation

AUCUN DOCUMENT ET AUCUNE MACHINE
NE PAS OUBLIER DE RENDRE L'ÉNONCÉ DANS VOTRE COPIE EN FIN D'ÉPREUVE !

NOM PRENOM		GROUPE	

1 Exercice 1 : notion de variable, type, affectation et de forçage de type (4 pts)

```
1  void algorithm() {
2    ... NB_LETTRES = 26;
3    ... alea     = random() * NB_LETTRES;
4    ... lettre   = ... ('a' + ... alea);
5    ... voyelle  = (lettre == 'a' || lettre == 'e' || lettre == 'i' ||
6                    lettre == 'o' || lettre == 'u');
7    ... points;
8    if (voyelle) {
9      points = 1;
10   } else {
11     points = 3;
12   }
13   String message = ... ;
14 }
```

1. (2) Indiquer les types de `NB_LETTRES`, `alea`, `lettre`, `voyelles`, `points` en expliquant pourquoi vous faites ce choix.
2. (2) Indiquer le code manquant pour l'initialisation de la variable `lettre` et de la variable `message` afin que cette dernière contienne la valeur `"XX point(s)."` avec à la place de XX la valeur de la variable `points`.

2 Exercice 2 : notion de condition, connecteur logique et d'alternative (5 pts)

1. (1) Ce tableau présente des expressions booléennes impliquant des variables dont les valeurs changent pour chaque colonne. Indiquer le résutat de l'évaluation de ces expressions en indiquant juste `T` pour `true` ou `F` pour `false`.

	`a = 18` `l = 'z'`	`a = 19` `l = 'a'`	`a = 42` `l = 'g'`
`(a < 19)`			
`(l >= 'g')`			
`(a < 19) && (l >= 'g')`			
`(a < 19) || (l >= 'g')`			
`(a >= 19) ^ (l >= 'g')`			

2. (2) Lire attentivement le programme ci-dessous et réaliser son organigramme.

```
1  void algorithm() {
2    // on suppose toutes les variables utilisées déjà déclarées et
     initialisées
3    if (!abonne) {
4       if (ticket == true && passager == 2) {
5         print("couple");
6       } else if (ticket && passager > 5)) {
7         print("grande famille");
8       } else {
9         print("famille");
10      }
11      println("!");
12   } else {
13     if (premium) {
14       print("cher.e ");
15     }
16     print("client.e)");
17     if (fidelite > 5) {
18       print("fidèle");
19     }
20   }
21 }
```

3. (2) Indiquer les affichages produits par le programme ci-dessus en fonction des valeurs proposées pour les différente variables dans le tableau ci-dessous.

Valeurs des variables	Affichage produit par le programme
abonne = true ticket = true passager = 2 premium = true fidelite = 2	
abonne = false ticket = true passager = 6 premium = false fidelite = 3	
abonne = true ticket = true passager = 2 premium = false fidelite = 6	
abonne = false ticket = true passager = 3 premium = true fidelite = 4	
abonne = true ticket = false passager = 1 premium = false fidelite = 0	

3 Exercice 3 : Débogage à la compilation et l'exécution (3 pts)

Voici un programme qui a été écrit peut-être un peu trop rapidement.

```
1  class Hum extends Program {
2    void algorithm() {
3      String phrase = readString();
4      boolean interrogative;
5      int Indice = 0
6      while (charAt(phrase, indice) != "?" || indice < length(phrase)) {
7        indice = indice + 1;
8      }
9      interrogative = (length(phrase) = indice);
10     println(phrase + "␣est␣interrogative␣:␣" + interrogative);
11   }
12 }
```

1. (1) Ce programme contient 3 erreurs de compilation : indiquez les et expliquez pourquoi cela produit une erreur.
2. (1) En supposant les erreurs de compilation corrigées, il y a 3 bugs de conception dans ce programme : indiquez à quelles lignes se trouvent l'erreur et pourquoi c'est une erreur.
3. (1) Donnez le programme valide ne contenant plus d'erreurs de compilation et conception.

4 Exercice 4 : notion de répétition et d'alternative (8 pts)

1. (2) Écrire le programme `NombreDePoints` qui compte le nombre de points présents dans une chaîne de caractères saisie au clavier.
2. (2) Écrire le programme `Email` qui affiche l'indice du premier caractère @ présent dans une chaîne de caractères saisie au clavier et affiche -1 sinon.
3. (2) Écrire un programme `SaisiePin` qui demande à l'utilisateur de saisir une suite de caractères représentant un code PIN (i.e. contenant exactement 4 chiffres). Le programme demande une saisie caractère par caractères jusqu'à avoir effectivement récupéré 4 chiffres.
4. (2) Écrire le programme `Play` permettant de produire les dessins donnés ci-dessous sans effectuer d'affichages inutiles (on suppose que l'on ne traite que les nombres impairs).

```
> ijava Play
N : 3
*
**
*
> ijava Play
N : 5
*
**
* *
**
*
> ijava Play
N : 7
*
* *
*   *
*     *
*   *
* *
*
```