**Correction TD1 : Analyse Syntaxique | Expressions régulières**

Exercice1 : **Opérations sur les Langages**

Soit deux ensembles : ***L = {A, B, ..., Z, a, b, ..., z}*** : ensemble des lettres de l'alphabet (majuscules et minuscules). ***C = {0, 1, ..., 9}*** : ensemble des chiffres.

À partir de (L) et (C), réponds aux questions suivantes :

1. Quel est le langage ***L ꓴ C*** ?

2. Quel est le langage ***L C*** (concaténation de L et C) ?

3. Décris ***L3*** (puissance du langage **(*L*)** en termes de chaînes générées.

4. Décris ***C+*** (un ou plusieurs chiffres) et donne des exemples de chaînes générées.

5. Décris ***L(L ꓴ C)\**** (lettre suivie d'une chaîne de lettres ou de chiffres ou chaîne vide) et donne quelques exemples.

|  |
| --- |
| **#### Correction Ex1 ####** |
| 1. Le langage ***L ꓴ C*** est l'ensemble des lettres (majuscules et minuscules) et des chiffres. Il contient des éléments tels que `A`, `b`, `5`.  2. Le langage ***L C,*** est l'ensemble des chaînes constituées d'une lettre suivie d'un chiffre, par exemple `A3`, `b5`, `Z9`.  3. Le langage ***L3*** est constitué de toutes les chaînes de trois lettres. Quelques exemples : `ABC`, `xyz`, `Aaz`, etc.  4. Le langage ***C+*** est l'ensemble des entiers naturels, soit toutes les chaînes d’un ou plusieurs chiffres. Exemples : `0`, `123`, `4567`.  5. Le langage ***L(L ꓴ C)\**** : représente les chaînes constituées d'une lettre suivie d'une séquence de lettres ou de chiffres (ou une chaîne vide). Quelques exemples : `A`, `B123`, `xZ9`, `a23`. |

Exercice 2 : **Opérations sur les langages**

Soit ***L1={a, b}*** et ***L2 = {0, 1}.***

1. Donne les éléments de ***L1 L2*** *(concaténation de* ***L1*** *et* ***L2****)*

2. Donne les éléments de ***L1*** *ꓴ* ***L2***

3. Décris ***L1***\* (fermeture de Kleene) et donne quelques exemples de chaînes générées.

4. Que représente l’expression ***L1***\* ***L2*** ?

|  |
| --- |
| **#### Correction Ex2 ####** |
| 1. Concaténation ***L1 L2*** : {***a0, a1, b0, b1}***  2. Union ***L1*** *ꓴ* ***L2*** : {a, b, 0, 1}  3. Fermeture de Kleene ***L1***\*: est l'ensemble des chaînes de longueur arbitraire (y compris la chaîne vide) générées à partir de ***L1***. Exemples : ``, `a`, `b`, `aa`, `ab`, `ba`, `bb`, etc.  4. Expression ***L1***\* ***L2***: est l'ensemble des chaînes commençant par une ou plusieurs lettres de L1 suivies d'un élément de ***L2*** Exemples : `a0`, `ab1`, `bb0`. |

Exercice 3 **: Langages et opérations**

Soit les langages suivants définis sur l'alphabet ***Σ= {a, b}*** : Soit ***L1={a,aa,b}*** et ***L2 = {b,bb,ab}.***

Effectue les opérations suivantes et donne les éléments du langage résultant :

1. Donne les éléments de ***L1*** *ꓴ* ***L2***

2. Donne les éléments de ***L1 L2*** *(concaténation de* ***L1*** *et* ***L2****)*

3. Décris ***L1***\* (fermeture de Kleene) et donne quelques exemples de chaînes générées.

|  |
| --- |
| **#### Correction Ex3 ####** |
| 1. Union ***L1*** *ꓴ* ***L2***: est l'ensemble des chaînes qui sont soit dans ***L1,*** soit dans *ꓴ* ***L2***, soit dans les deux.  ***L1*** *ꓴ* ***L2***= {a, aa, b, bb, ab}  2. Concaténation ***L1******L2***: consiste à prendre chaque élément de ***L1*** et à le concaténer avec chaque élément de ***L2***.  ***L1L2***= {ab, abb, aab, aabb, bbb, bab}  3. Fermeture de Kleene ***L1\**** : est l'ensemble de toutes les concaténations possibles de zéro ou plusieurs éléments de ***L1***, incluant la chaîne vide ***ε***. Quelques exemples :  ***L1\* ={ ε, a, aa, b, aaa, aab, aaab, aaaa, bb, aaaa, aba, ...}*** |

Exercice 4 : **Langage et expressions régulières**

Donner une expression régulière +pour chacun des langages suivants sur l'alphabet ***L ꓴ C*** (lettres et chiffres) :

1. Les chaînes qui commencent par une lettre majuscule suivie d'une ou plusieurs lettres minuscules.

2. Les chaînes qui contiennent exactement trois chiffres.

3. Les chaînes qui commencent par une lettre, suivie d'une séquence de lettres et de chiffres, et qui se terminent par un chiffre.

|  |
| --- |
| **#### Correction Ex4 ####** |
| 1. Chaîne commençant par une majuscule, suivie de minuscules :  Expression régulière : [A-Z][a-z]+    2.Chaîne contenant exactement trois chiffres\*\* :  Expression régulière : [A-Za-z]\*[0-9][A-Za-z]\*[0-9][A-Za-z]\*[0-9][A-Za-z]\*  3. Chaîne commençant par une lettre, suivie d'une séquence de lettres/chiffres et se terminant par un chiffre :  Expression régulière : [A-Za-z][A-Za-z0-9]\*[0-9] |

Exercice5 : **Lexèmes sur un alphabet**

Donner une expression régulière qui décrit les lexèmes sur l'alphabet `{a, b, c}` qui commencent par `b`.

|  |
| --- |
| **#### Correction Ex5 ####** |
| b(a|b|c)\* |

Exercice 6 : **Expressions régulières**

Écris des expressions régulières pour les langages suivants :

1. Les chaînes contenant au moins un `a`.

2. Les chaînes contenant un nombre pair de `a` (peu importe le nombre de `b`).

3. Les chaînes qui commencent par `a`, contiennent un nombre quelconque de `b`, et finissent par `a`.

|  |
| --- |
| **#### Correction Ex6 ####** |
| 1. Chaînes contenant au moins un `a` :  Expression régulière : ***(b\*a)+***  Cela signifie qu'il peut y avoir n'importe quel nombre de `b` (ou zéro), suivi d'au moins un `a`.  2. Chaînes contenant un nombre pair de `a` : (***b\*ab\*ab\*) \****  Cette expression représente un nombre pair de `a`, séparés ou non par des `b`. Il peut y avoir zéro ou plusieurs occurrences de cette structure.  3. Chaînes commençant par `a`, contenant des `b` et finissant par `a` :  Expression régulière : a(b\*)a |

Exercice 7 : **Langages et expressions régulières avancées**

Donne une expression régulière pour les langages suivants :

1. Les chaînes sur l'alphabet {a, b} contenant un nombre pair de `a`.

2. Les chaînes sur l'alphabet ***{a, b}*** contenant exactement deux `b`.

3. Les chaînes sur l'alphabet ***{0, 1}*** qui ne contiennent pas le sous-motif `00`.

|  |
| --- |
| **#### Correction Ex7 ####** |
| * Il y a un nombre pair de `a` séparés par des `b`. (b\*ab\*ab\*)\* * Exactement deux `b` : a\*ba\*ba\* * Ne contient pas `00` (1|01)\*0? |

Exercice 8 **: Langage de mot de passe**

Écris une expression régulière pour un mot de passe valide respectant les règles suivantes : Contient au moins 8 caractères, doit inclure au moins une lettre majuscule, une lettre minuscule, et un chiffre, ne doit pas contenir d'espace.

|  |
| --- |
| **#### Correction Ex8 ####** |
| (?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)[A-Za-z\d]{8,} |

Exercice 9 **: Utilisation des ancrages**

**Énoncé** : Tu es en train de développer un compilateur, et tu souhaites valider des identifiants de variables. Ces identifiants doivent :

1. Commencer par une lettre (majuscule ou minuscule) ou un underscore (\_).
2. Être suivis par n'importe quel nombre de lettres, chiffres ou underscores.

Écris une expression régulière utilisant des **ancrages** pour valider les identifiants

|  |
| --- |
| **#### Correction Ex9 ####** |
| ^[a-zA-Z\_][a-zA-Z0-9\_]\*$  - `^` : ancrage de début de ligne (l'identifiant doit commencer ici).  - `[a-zA-Z\_]` : l'identifiant doit commencer par une lettre majuscule, une lettre minuscule ou un underscore.  - `[a-zA-Z0-9\_]\*` : après le premier caractère, il peut y avoir zéro ou plusieurs lettres, chiffres ou underscores.  - `$` : ancrage de fin de ligne (l'identifiant doit se terminer ici). |

Exercice 10 : **Utilisation des assertions**

Tu développes un analyseur lexical qui doit détecter si une variable est suivie d'un opérateur d'affectation (=) sans espace, et renvoyer une erreur si c'est le cas. La variable suit les mêmes règles que dans l'exercice précédent. Écris une expression régulière avec une **assertion négative** qui s'assure que la variable n'est **pas** suivie immédiatement de =.

|  |
| --- |
| **#### Correction Ex10 ####** |
| ^[a-zA-Z\_][a-zA-Z0-9\_]\*(?!\=).\*  `^[a-zA-Z\_][a-zA-Z0-9\_]\*` : correspond à une variable valide, comme dans l'exercice précédent.  - `(?!\=)` : assertion négative, qui garantit que le prochain caractère n'est pas un signe égal (`=`). Cela empêche les correspondances avec les variables suivies d'un opérateur d'affectation sans espace.  - `.\*` : après la variable, il peut y avoir d'autres caractères, mais ils ne sont pas immédiatement après la variable. |