**Exercice 1 :**

Les grammaires qui génèrent les langages suivants :

**1)** L’ensemble des nombres binaires (un langage infini qui ne contient pas le mot vide)

**Exemple :** L = {0, 1, 01, 10, 00, 11, 001, 111, 00000, 10010, 00110, … }

G=<T, N, S, P> où T={0,1} N={S} et P=

**S🡪 0S / 1S / 0 / 1**

**2)** L’ensemble des nombres binaires sans 0 inutiles en tête

**Exemple :** L={0, 1, 10, 101, 100110, 111, 100000, … }

G=<T, N, S, P> où T={0,1} N={S, A} et P=

**S🡪 1A / 0**

**A🡪 0A / 1A / ɛ**

ou bien

G=<T, N, S, P> où T={0,1} N={S,A} et P=

**S🡪 A / 0**

**A🡪 A0 / A1 / 1**

**3)** L’ensemble des nombres binaires de longueur paire.

**Exemple :** L = {00, 01, 10, 11, 1001, 0010, 1111, 0000, 100101, 111111, … }

G=<T, N, S, P> où T={0,1} N={S, A} et P=

**S🡪 1A / 0A**

**A🡪 0S / 1S / 0 / 1**

ou bien

G=<T, N, S, P> où T={0,1} N={S} et P=

**S🡪 00S / 01S / 11S / 10S / 00 / 01 / 11 / 10**

**4)** Les nombres décimaux éventuellement signés n’ayant pas de 0 inutiles. Rappelons que la partie (optionnelle) après la virgule ne se termine pas par un 0.

**Exemple :** L = {0, 2020, +123.45, -67.89, 0.98, +0.14, 987.65, +4321, … }

G=<T, N, S, P> où T= {0, …, 9} N={S, A, B, D} et P=

**S🡪 A / +A / -A**

**A🡪 1B / 2B/ … / 9B / 0 / 0.D**

**B🡪 0B / 1B / 2B/ … / 9B / .D / ɛ**

**D 🡪 0D / 1D/ … / 9D / 1 /…/ 9**

**5)** L’ensemble des noms de variables (identificateurs) en Java. Un nom de variable en Java commence par une lettre alphabétique ou le caractère underscore (\_) suivi par une suite quelconque de lettres alphabétiques, de chiffres et l'underscore.

Exemple : L={a, nom, nom\_pere, p1, p2, \_prix, \_16\_04\_2020, thl\_2020, …..}

G=<T, N, S, P> où T ={a, ….,z , A,….,Z, 0,….,9, \_},

N={<NomJava>, <Suite>, <Lettre>, <Chiffre>} et P est :

<NomJava> → < Lettre > <Suite> / \_<Suite>

<Suite> → <Lettre> <Suite> / <Chiffre> <Suite> / \_<Suite> / ε

**<Lettre> → a /…. / z / A /…. / Z**

<Chiffre> → 0 /…. / 9

**6)** L'ensemble des tableaux de caractères alphabétiques. Un tableau commence par { et se termine par } et les caractères sont séparés par une virgule. Chaque caractère est compris entre deux quotes simples. Le tableau peut être vide.

**Exemple :** L= {{}, {**′**a**′**}, {**′**b**′, ′**c**′, ′**d**′** }, …}

G=<T, N, S, P> où T = {a, …, z, A, …, Z, **′, \**{, \}, \, }

N={<Tableau>, <SuiteCar>, <Car>} et S= <Tableau>, P=

**<Tableau> 🡪 { <SuiteCar> } / { }**

**<SuiteCar> 🡪 ′<Car>′, <SuiteCar> / ′Car′**

**<Car> 🡪 a / b / … / z / A / …/ Z**

**7)** L’ensemble des mots de passe de sécurité faible, qui sont formés que des lettres ou que des chiffres.

**Exemple :** L = {a, 7, aa, 2020, 04, thl, usthb, 16, … }

G=<T, N, S, P> où T={0,..9, a,…z, A, …, Z}

N={{<Password>, <SuiteLettres>, <SuiteChiffres>, <Lettre>, <Chiffre>} et

S= <Password> et P =

**<Password>🡪 <SuiteLettres> / <SuiteChiffres>**

**<SuiteLettres>🡪 <Lettre> <SuiteLettres> / <Lettre>**

**<SuiteChiffres>🡪 <Chiffre> <SuiteChiffres> / <Chiffre>**

**<Lettre>🡪a / b /…../ z / A / B /…../ Z**

**<Chiffre>🡪0 / 1 /…. / 9**

**8)** L’ensemble des mots de passe de sécurité moyenne, qui comportent au moins une lettre **et** au moins un chiffre mais aucun caractère spécial.

**Exemple :** L = {a7, 9aa, 20z20, s400, thl2020, usthb16alger, … }

G=<T, N, S, P> où T={0,..9, a,…z, A, …, Z}

N={<Password>, <Suite>, <Chiffre>, <Lettre> } et S= <Password>

L’ensemble des productions =

**<Password> 🡪<Suite> <Chiffre> <Suite> <Lettre> <Suite> /**

**<Suite> <Lettre> <Suite> <Chiffre> <Suite**

**<Suite>🡪<Lettre> <Suite> / <Chiffre> <Suite> / ɛ**

**<Chiffre> 🡪 0 / 1 / … / 9**

**<Lettre>🡪 a / b /…../ z / A / B /…../ Z**

érentes possibilités d’une lettre\*/

**Exercice 2 :**

Il faut remarquer que chaque mot correspondant à un message est composé de deux parties :

**Exemples de messages valides (respectant la parité)**

L={00, 11, 000, 101, 10111, 10100, 110101, ….}

**Exemples de messages altérés (ne respectant pas la parité)**

L={01, 10, 001, 100, 10110, 10101, 110100, ….}

Finalement, l’ensemble des mots respectant la parité est le langage des mots ayant un nombre pair de 1. Sa grammaire est :

G=<T, N, S, P> où T={0,1} N={S,P, I} et les productions :

**S🡪 0P / 1I** /\* 0 ou 1 est le plus petit message qu’on lui ajoutera le bit de parité\*/

**P🡪 0P / 1I / 0** /\* P représente le nombre pair de 1 \*/

**I🡪 0I / 1P/ 1** /\* I représente le nombre impair de 1 \*/