Les trois parties sont indépendantes ; chacune sera rédigée sur une copie séparée (plusieurs au besoin).

Partie 1 : logique

▶ On dispose d'un ensemble $\mathcal{V} = \{x, y, z, \ldots\}$ de *variables logiques*. Un *littéral* est une variable x ou sa négation $\neg x$. Une *clause* est un littéral ou une disjonction de littéraux : $x \vee z$, $\neg y$ et $(\neg x) \vee y \vee z$ sont des clauses.

▶ Une *assignation* est une fonction φ de $\mathcal{V}$ dans l'ensemble $\{\text{vrai}, \text{faux}\}$. Les tableaux suivants permettent d'étendre φ aux littéraux, puis aux clauses :

$\varphi(x)$	$\varphi(\neg x)$
faux	vrai
vrai	faux

$\varphi(c)$	$\varphi(c')$	$\varphi(c \vee c')$
faux	faux	faux
faux	vrai	vrai
vrai	faux	vrai
vrai	vrai	vrai

▶ Une assignation φ *satisfait* une clause c si $\varphi(c) = \text{vrai}$.

Question 1 Énumérer les assignations qui satisfont simultanément toutes les clauses de l'ensemble E suivant :

$$E = \{x \vee (\neg y) \vee z, (\neg x) \vee (\neg y) \vee z, (\neg x) \vee y \vee (\neg z), x \vee y \vee (\neg z), (\neg x) \vee (\neg y) \vee (\neg z)\}$$

Question 2 Déterminer le nombre maximal de clauses que l'on peut satisfaire simultanément, dans l'ensemble F suivant :

$$F = \{x, y, z, w, (\neg x) \vee (\neg y), (\neg y) \vee (\neg z), (\neg z) \vee (\neg x), x \vee (\neg w), y \vee (\neg w), z \vee (\neg w)\}$$

Partie 2 : automates finis

▶ Σ désigne l'alphabet $\{a, b\}$. Soient $x \in \Sigma$ et $u \in \Sigma^*$; $|u|_x$ est le nombre d'occurrences de la lettre x dans le mot u. Notons L l'ensemble des mots qui contiennent au moins 3 occurrences de la lettre a : $u \in L \iff |u|_a >= 3$.

Question 3 Donner une expression rationnelle décrivant L.

Question 4 Décrire un automate fini reconnaissant L.

▶ Notons M l'ensemble des mots qui contiennent plus d'occurrences de a que de b :

$$u \in M \iff |u|_a > |u|_b$$

Question 5 Le langage M est-il rationnel ?

Question 6 Le langage $M \setminus L = \{u \in M \mid u \notin L\}$ est-il rationnel ?

Partie 3 : arbres binaires et programmation

▶ Nous nous intéressons dans cette partie à des arbres binaires, dont les feuilles sont étiquetées par des entiers. Voici un exemple :

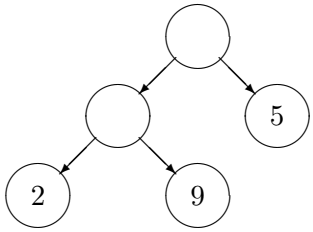

FIG. 1 – un arbre binaire dont les feuilles sont étiquetées par des entiers

▶ La *profondeur* d'une feuille d'un arbre a est la longueur du chemin qui mène de la racine de a à cette feuille. Ainsi, dans l'arbre de la figure 1, l'une des feuilles est à la profondeur 1, et les deux autres à la profondeur 2.

Question 7 Soit a un arbre binaire ; notons $p(a)$ la profondeur minimale d'une feuille de a. Montrer que a contient au moins $2^{p(a)}$ feuilles.

Programmation en Caml

Cette partie ne s'adresse qu'aux étudiants qui programment en langage Caml.

▶ Pour décrire les arbres binaires de ce problème, nous définissons le type Caml suivant :

```
type arbre = N of arbre * arbre | F of int ;;
```

Par exemple, `let a = N(N(F 2,F 9),F 5) ;;` décrit l'arbre de la figure 1.

Question 8 Rédiger en langage Caml une fonction de signature :

```
etiquette_maximale : arbre -> int
```

spécifiée comme suit : `etiquette_maximale a` calcule la valeur maximale d'une étiquette d'une feuille de l'arbre a.

Question 9 Rédiger en langage Caml une fonction de signature :

```
profondeur_minimale : arbre -> int
```

spécifiée comme suit : `profondeur_minimale a` calcule la profondeur minimale d'une feuille de l'arbre a.

Programmation en Pascal

Cette partie ne s'adresse qu'aux étudiants qui programment en langage Pascal.

▶ Pour décrire les arbres binaires de ce problème, nous définissons le type Pascal suivant :

```
type
 ptrArbre = ^Arbre;
 Arbre = record
    n_ou_f : (noeud,feuille);
    gauche, droit : ptrArbre;
    etiquette : integer;
   end;
```

Si la valeur du champ `n_ou_f` est `noeud`, alors les champs `gauche` et `droit` donnent les fils gauche et droit, et le champ `etiquette` n'a pas de signification. Si par contre la valeur du champ `n_ou_f` est `feuille`, alors le champ `etiquette` donne l'étiquette, et les champs `gauche` et `droit` n'ont pas de signification.

Question 8 Rédiger en langage Pascal une fonction d'en-tête :

```
function etiquette_maximale(p:ptrArbre):integer;
```

spécifiée comme suit : `etiquette_maximale(a)` calcule la valeur maximale d'une étiquette d'une feuille de l'arbre a.

Question 9 Rédiger en langage Pascal une fonction d'en-tête :

```
function profondeur_minimale(p:ptrArbre):integer;
```

spécifiée comme suit : `profondeur_minimale(a)` calcule la profondeur minimale d'une feuille de l'arbre a.

FIN