Travail Ecrit

Nom et Prénom : Zambaz Yannick Classe : I2a

Barème	(1)	(2)	(3)	(4)	Total
	16 pts	8 pts	18 pts	8 pts	50
	16	8	16	2	42

5.4

1. Définir, sur l'alphabet {a, b} le langage des mots contenant un *nombre pair* de 'a' *et exactement deux* 'b', au moyen de :

   a) un automate d'état fini déterministe;
   b) une expression régulière.

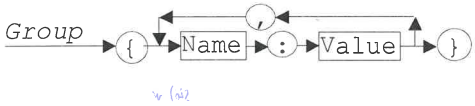
   (a|a)* b (aa*)* b (aa)*
   ou bba

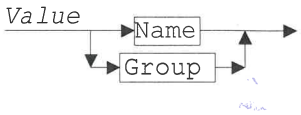
2. Faire évoluer l'assertion suivante de ligne en ligne.

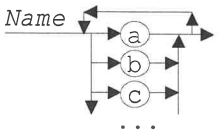
```
int a,b;
...
          // a == 9 - 3*b
b++;
          // ...

a=a/3;
          // ...
```

3. Les diagrammes syntaxiques suivants définissent un langage (proche de la syntaxe JSON) pour décrire des propriétés.







On veut écrire un programme qui convertit une telle description en une structure de données contenant toutes les propriétés; voici un exemple pour illustrer (observer l'imbrication *x.y...*) :

```
{
  model : fiesta,
  option: { color:blue, airbag:yes },
  discount: no
}
```

Map (set of key-value pairs):

```
          "model" -> "fiesta"
       "discount" -> "no"
   "option.color" -> "blue"
"option.airbag" -> "yes"
```

a) Expliquer quel sera le rôle de l'*analyseur lexical* dans cette application.

b) Déclarer une classe avec tous les entêtes de méthodes publiques pour un *analyseur lexical* (juste la spécification, pas l'implémentation).

```
class Lexer {
  //...
}
```

c) Ecrire un interpréteur qui effectue la conversion demandée, en utilisant votre analyseur lexical.

```
public class Interpreter {
  public static MyMap decode(String wholeText) throws Exception {
    Lexer lex = new Lexer(wholeText);
    MyMap map = new MyMap();
    //...
  }
  //...
}
```

```
public class MyMap {
  public put(String key, String value);
  public boolean containsKey(String key);
  public String          get(String key);
}
```

4. Soit *H* le problème de décision suivant :

   INPUT : une machine de Turing *T* qui retourne un booléen (TRUE si l'input est accepté), et dont l'exécution nécessite un nombre d'étapes *polynomial* par rapport à son input.

   Existe-t-il une donnée que la machine *T* accepte ?

   pas de variable?

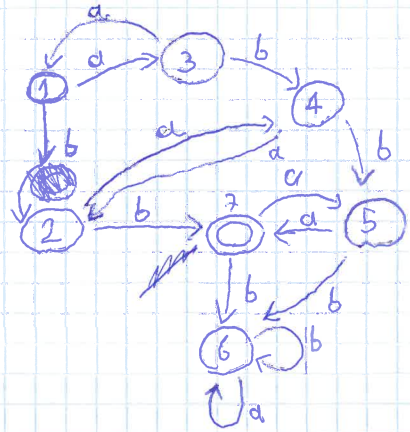
En justifiant/argumentant vos réponses, expliquer ce qu'on peut dire du problème *H*, Est-il décidable, appartient-il (oui, non, peut-être) à NP, à P, et à NP-Complet ?

Zambon, Yannick
T2-a

TE01 Langage Formel

Exercice 1

a)



1 : a pair 0 b
2 : a pair 1 b
3 : a impair 0 b
4 : a impair 1 b
5 : a impair 2 b
6 : >2 b
7 : a pair 2 b

b) $((aa)^* b (aa)^* b (aa)^* \cup a(aa)^* b a(aa)^* b(aa)^* \cup a(aa)^* b (aa)^* b a(aa)^* \cup (aa)^* b a(aa)^* b a(aa)^*)$ ✓

abba ✓ bb ✓ baaab ✓ aabaab ✓ → Seems OK.

Exercice 2

```
int a, b;        // a == 9 - 3*b
b++;             // a == 9 - 3*(b-1)
                 // a == 9 + 3 - 3*b
                 // a == 12 - 3*b
a = a/3          // 3a == 12 - 3*b
                 // → a == 4 - b
```

Exercice 3

L'analyseur lexical a pour but de vérifier la structure de donnée soit correcte syntaxiquement. Le lexer découpe la structure en "Symbole" et les parcourt un à un ✓ (2)

a) ~~L'analyseur lexical va vérifier que la structure de données soit correcte syntaxiquement (indépendamment des données). Le lexer va tout que~~

b)
```
class Lexer {
    public String currentSymbol;   // attribut pointeur
    public Lexer (String a);       // constructeur
    public void goToNextSymbol();  // bouger le pointeur.
    public boolean endOfString();  // teste si le String est terminé.
    public String getNextWord();   // move and return the next char
                                   // until a special char is encounter.
                                   // (or a space)
    public String checkNextSymbol(); // return the next symbol without moving.
```

(3)

c)

```
public class Interpreter {
    public static Lexer lex;
    public static MyMap decode (String wholeText) throws Exception {
        lex = new Lexer (wholeText);
        MyMap map = new MyMap();
        map = parseGroup (map);
        if (!lex.endOfString()) { throw Exception(); }
        return map; }

    public static MyMap parseGroup (MyMap map) throws Exception {
        if (lex.currentSymbol != '{') { throw Exception(); }
        lex.goToNextSymbol();
        String name = parseName();
        if (lex.currentSymbol != ':') { throw Exception(); }
        lex.goToNextSymbol();
        String value = parseValue();
                    // work only if there is no nested Group...
```

```
public class Interpreter {
    public static Lexer lex;
    public static MyMap map;
    public static void decode (String wholeText) throws Exception {
        Lexer lex = new Lexer (wholeText);
        MyMap map = new MyMap();
        parseGroup();
        if (!lex.endOfString()) { throw Exception(); }
    }
    public static parseGroup () throws Exception {
        if (lex.currentSymbol != '{') { throw Exception(); }
        lex.goToNextSymbol();
        do {
            String name = parseName();
            if (lex.currentSymbol != ':') { throw Exception(); }
            String value = parseValue();
            map.put (name, value);
        } while (lex.currentSymbol == ',');
        lex.goToNextSymbol();
        if (lex.currentSymbol() != '}') { throw Exception(); }
    return ;
}
```

✓

✓

fails for nested groups like option.color ÷2

Zamboa Yannick

T2-a

```
public static String parseName() {
    return lex.getNextWord();}

public static String parseValue() {
    if (lex.checkNextSymbol() == '{')
        return parseGroup();    //don't work
    else {return name;}
}
}
```

✓

✓

← // ici le branchement ne fait penser que l'exercice n'est pas possible ...

4. Si une machine de Turing peut résoudre un problème en un temps polynomial, le problème est forcément décidable et appartient au groupe P. (+2)

Mais j'avoue ne même pas comprendre la donnée 😕.

- ~~Possibilités d'interprétations des problèmes :~~
  - ~~C'est une fonction qui prend en entrée une machine de Turing polynomial ;~~
    - ~~Alors, la fonction est décidable, il y a une réponse en temps polynomial.~~
    - ~~La fonction est peut-être P résolu en un temps P~~
- ~~Interprétation : Le problème~~