Correction de l'exercice 1

Question 1 : *Calcul des R_i définis par :*

$$\left\{\begin{array}{lcl} R_0 & = & \{f/\exists u \in C, uf \in C\} \\ R_i & = & \{f/\exists u \in R_{i-1}, uf \in C\} \cup \{f/\exists u \in C, uf \in R_{i-1}\} \end{array}\right.$$

$$C = \{00, 01, 011, 1110\}$$
$$R_0 = \{1\}$$
$$R_1 = \{110\} \cup \{\}$$
$$R_2 = \{\} \cup \{\}$$
$$R_i = \emptyset \textit{ pour } i > 1$$

Cette application définit un code car : $\forall i, R_i \cap C = \emptyset$.
Le code est à delai borné car : $\forall i > 1, R_i = \emptyset$.

Question 2 :

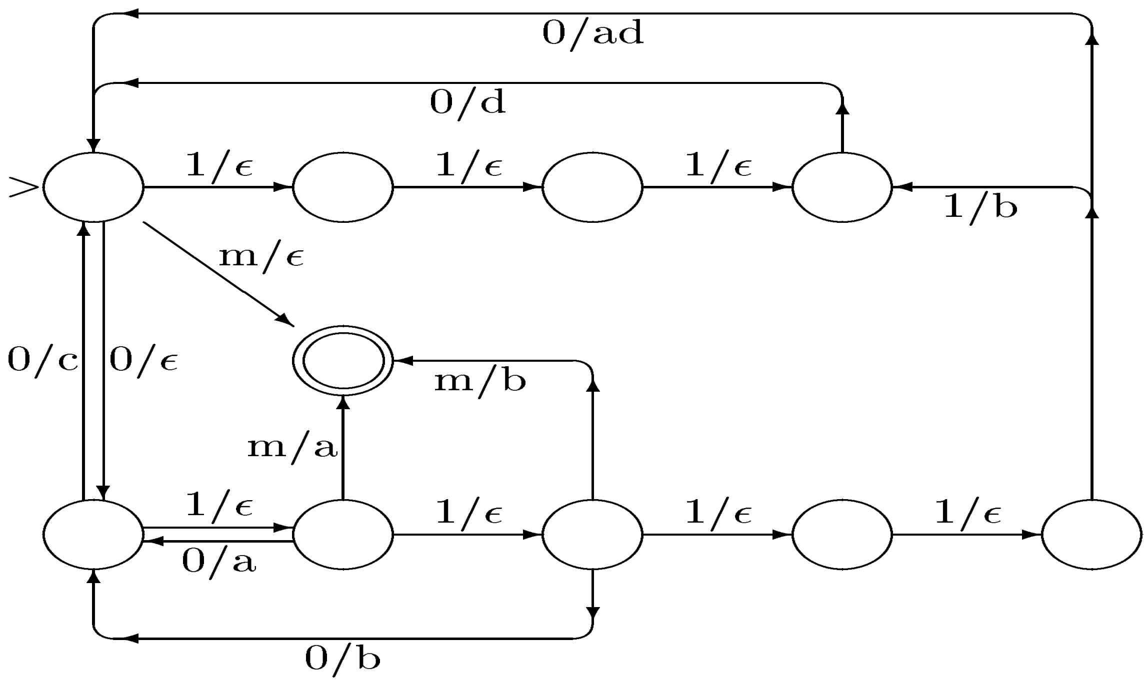

Figure 1: automate avec sortie pour décoder ϕ

Question 3 : *Un programme lex possible pour decoder ϕ.*

```
%option noyywrap
%{
%}
```

```
%%

01/(0|m)  {printf("a");}
011/(0|m) {printf("b");}
00        {printf("c");}
1110      {printf("d");}
011110    {printf("ad");}
0111110   {printf("bd");}
m         {return;}
.         {printf("Error");}

%%

main()
{
  yylex();
}
```

Correction de l'exercice 2

Question 1 : *Un programme lex possible pour compter les constantes entières d'un texte.*

```
%option noyywrap
%{
  int nbConstantesEntieres = 0;
%}

%%

[0-9]+   {nbConstantesEntieres++;}
\n       {}
.        {}

%%

main()
{
  yylex();
  printf("Nombre de constantes entières : %d\n", nbConstantesEntieres);
}
```

Correction de l'exercice 3

Question 1 :

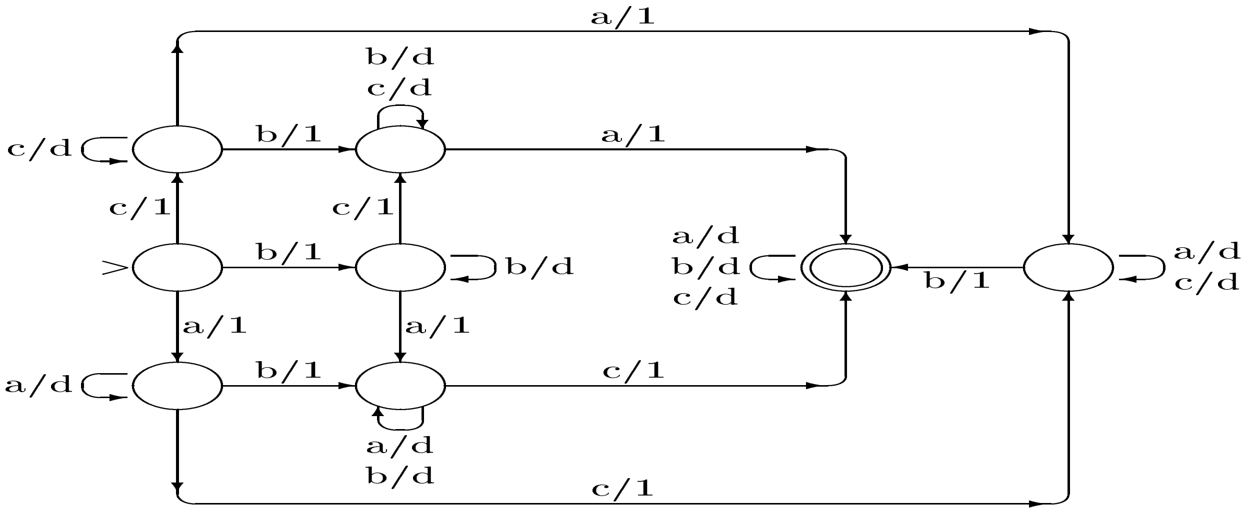

Figure 2: automate avec sortie : première occurence → 1, autres → d

Question 2 :
La première est un 1 *et il y a un maximum de trois* 1.

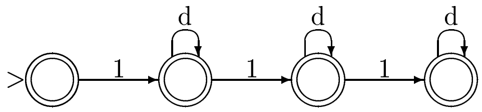

Figure 3: l'ensemble des mots $1d^*(1d^*)?(1d^*)?$

Question 3 :
Il faut au minimum 27 états. Un état code les nombres d'occurences ($0, 1$ *ou* ≥ 2) *pour chacune des lettres* a, b *et* c. *il y a donc* $3^3 = 27$ *états.*

Question 4 :

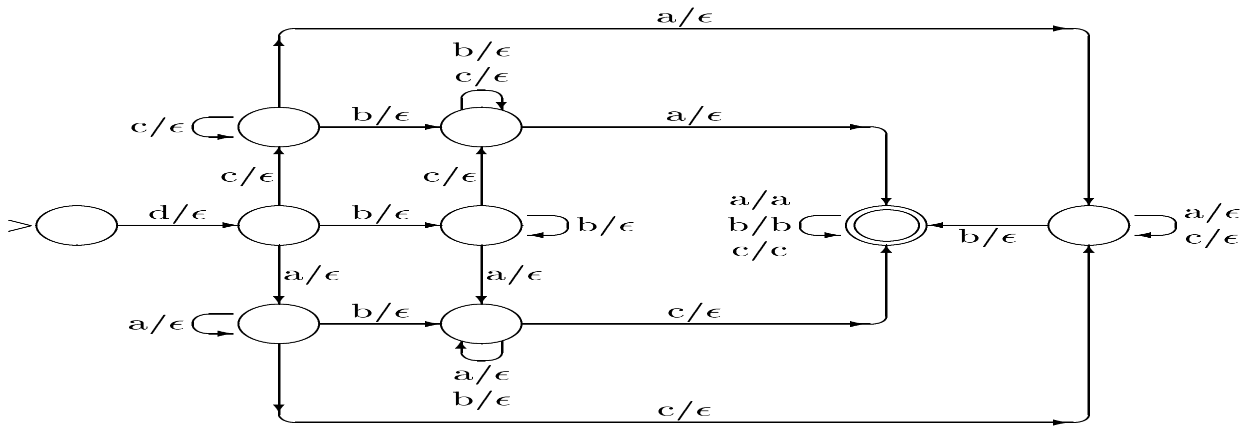

Figure 4: automate avec sortie : suffixe après une occurence de a,b et c

Question 5 :

Un programme lex possible pour decoder ψ.

```
/* les suffixes des mots ayant au moins 1a, 1b et 1c */

%option noyywrap

%%

da[a]*b[a|b]*c   {}
da[a]*c[a|c]*b   {}
db[b]*a[a|b]*c   {}
db[b]*c[b|c]*a   {}
dc[c]*a[a|c]*b   {}
dc[c]*b[b|c]*a   {}
a                {printf("a");}
b                {printf("b");}
c                {printf("c");}

.                {printf("Error");}

%%
main()
{
  yylex();
}
```