Remarques :

- L'option `-v` de lex permet d'observer le travail effectué.
- L'option `-T` de lex permet d'obtenir l'automate exécuté.

Correction de l'exercice 1

```
/* remplace 0 par a si suivi par 0 par b sinon. */
/* remplace 1 par c si suivi par 1 par b sinon. */

%option noyywrap

%%

0/0      {printf("a");}
0/[^0]   {printf("b");}
1/1      {printf("c");}
1/[^1]   {printf("b");}

.   {printf("Error\n");return;}

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

Correction de l'exercice 2

Une solution avec des états :

```
/* remplace 0 par a si precede par 0 par b sinon. */
/* remplace 1 par c si precede par 1 par b sinon. */

%option noyywrap

%x PrecedePar0 PrecedePar1

%%

<INITIAL>0     {printf("b");BEGIN(PrecedePar0);}
<INITIAL>1     {printf("b");BEGIN(PrecedePar1);}
<PrecedePar0>0 {printf("a");}
<PrecedePar0>1 {printf("b");BEGIN(PrecedePar1);}
<PrecedePar1>0 {printf("b");BEGIN(PrecedePar0);}
<PrecedePar1>1 {printf("c");}
```

```
<*>\n          {printf("\n");BEGIN(INITIAL);}
<*>.           {printf("Error\n");return;}

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

et une basée sur la solution de l'exercice précédent :

```
/* remplace 0 par a si precede par 0 par b sinon. */
/* remplace 1 par c si precede par 1 par b sinon. */

%option noyywrap

%%

^0/0 {printf("ba");}
^0/1 {printf("bb");}
^1/1 {printf("bc");}
^1/0 {printf("bb");}
0/0  {printf("a");}
0/1  {printf("b");}
1/1  {printf("c");}
1/0  {printf("b");}
^0$  {printf("b");}
^1$  {printf("b");}
0$   {}
1$   {}

.    {printf("Error\n");return;}

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

Correction de l'exercice 3

```
/* décodage de phi(a)=0 phi(b)=01, phi(c)=11. */

%option noyywrap

C       11
```

```
NbPair1 {C}*

%%

0/{NbPair1}  {printf("a");}
01/{NbPair1} {printf("b");}
{C}          {printf("c");}

.            {printf("Error\n");return;}

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

Correction de l'exercice 4

```
/* les identificateurs d'un programme java */

%option noyywrap

KEYWORDS abstract|boolean|break|byte|case|catch|char|class|continue|default|do|double|else|extends|false|final|finally|float|for|if|implements|import|instanceof|int|interface|long|native|new|null|package|private|protected|public|return|short|static|super|switch|synchronized|this|throw|throws|transient|true|try|void|volatile|while

NEWLINE           "\n"
IDENT             [a-zA-Z$]([a-zA-Z$_0-9]*".")*[a-zA-Z$_0-9]+
COMMENTLIGNEDEBUT "//"
COMMENTLIGNE      {COMMENTLIGNEDEBUT}.*{NEWLINE}
COMMENTDEBUT      "/*"
COMMENTFIN        "*/"|"**/"
MOTIF1            {COMMENTDEBUT}[^*]*{COMMENTFIN}
MOTIF2            {COMMENTDEBUT}([^*]*[*]+[^/])*[^*]*{COMMENTFIN}

COMMENT           {MOTIF1}|{MOTIF2}
%%

{KEYWORDS}|{NEWLINE} {}
{IDENT}              {printf("%s\n",yytext);}
{COMMENTLIGNE}       {}
{COMMENT}            {}

<*>.                 {}

%%
main ()
{
```

```
  yylex();
}
```

Correction de l'exercice 5

```
/* 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();
}
```

Correction de l'exercice 6

```
/* les identificateurs d'un programme java classés alphabétiquement */

%option noyywrap
%{
int niveauComment = 0;
char **idents;
int nbIdents = 0;
%}

%x COMMENTLIGNE COMMENTIMBRIQUE

KEYWORDS abstract|boolean|break|byte|case|catch|char|class|continue|default|do|double|else|extends|false|final|finally|float|for|if|implements|import|instanceof|int|interface|long|native|new|null|package|private|protected|public|return|short|static|super|switch|synchronized|this|throw|throws|transient|true|try|void|volatile|while

NEWLINE             "\n"
```

```
DIGIT                [0-9]
ENTIER               {DIGIT}+
DECIMALE             "."{DIGIT}*
SUFFIXE              E{ENTIER}
REEL                 {ENTIER}{DECIMALE}?{SUFFIXE}?
IDENT                [a-zA-Z$]([a-zA-Z$_0-9]*(".."))*[a-zA-Z$_0-9]*
COMMENTLIGNEDEBUT    "//"
COMMENTIMBRIQUEDEBUT "/*"
COMMENTIMBRIQUEFIN   "*/"

%%

<INITIAL>{KEYWORDS}|{NEWLINE}|{DIGIT}|{ENTIER}|{REEL} {}
<INITIAL>{IDENT} {
  nbIdents++;
  idents = realloc(idents,nbIdents*(sizeof (char *)));
  idents[nbIdents-1]=strcpy(malloc(strlen(yytext)),yytext);
}
<INITIAL>{COMMENTLIGNEDEBUT}            {BEGIN(COMMENTLIGNE);}
<INITIAL>{COMMENTIMBRIQUEDEBUT}         {niveauComment++;BEGIN(COMMENTIMBRIQUE);}

<COMMENTLIGNE>{NEWLINE}                 {BEGIN(INITIAL);}

<COMMENTIMBRIQUE>{COMMENTIMBRIQUEDEBUT} {niveauComment++;}
<COMMENTIMBRIQUE>{NEWLINE}              {}
<COMMENTIMBRIQUE>{COMMENTIMBRIQUEFIN}   {
  niveauComment--;
  if (niveauComment==0){
    BEGIN(INITIAL);
  }
}

<*>. {}

%%
void afficher(char **liste, int taille)
{
  char **aux;
  for (aux=liste; aux < liste + taille; aux++)
    printf("%s\n",*aux);
}

int compare(void **ch1, void **ch2){
  return strcmp(*ch1,*ch2);
}

main ()
```

```
{
  yylex();
  qsort(idents,
        nbIdents,
        sizeof (char *),
        (int (*)(const void *, const void *))compare);
  afficher(idents,nbIdents);
}
```