Flex

Correction de l'exercice 1 :

```
%%
^" "*
" "*         { printf(" ") ; }
%%
int main()
{
  yylex() ;
  return 0 ;
}
```

Correction de l'exercice 2 :

```
chf [0-9]+
%{
int mots = 0, signes = 0, nombres = 0 ;
%}
%%
[-a-zA-Z]+                    {mots++ ;}
[,. : ! ? ;-]                 {signes++ ;}
{chf}(" "{chf})*(","{chf})?   {nombres++ ;}
.
\n
%%
...
```

Correction de l'exercice 3 :

```
%%
begin|end               {printf("mot-clef %s\n",yytext) ;}
[a-zA-Z][a-zA-Z0-9]*    {printf("identificateur %s\n",yytext) ;}
[0-9]+                  {printf("nombre %i\n",atoi(yytext)) ;}
\+|\-|\*\*|\*           {printf("operateur %s\n",yytext) ;}
[ \t\n]+
.                       {printf("erreur %s\n",yytext) ;}
%%
...
```

Correction de l'exercice 4 :

```
...
\"([^\"\\]* (\\.)?(\\\n)?)* \"   {printf("chaine %s\n",yytext) ;}
...
```

Correction de l'exercice 5 :

```
...
"/*"  ([^*]*("*"+[^*/])*)*  "*"+"/"   {printf("commentaire\n") ;}
...
```

Correction de l'exercice 6 :

```
%{
int com = 0;
%}
%x commentaire
%%
...
"/"+"*"              { printf("commentaire\n"); com++; BEGIN(commentaire); }

<commentaire>"/"+"*"  { com++; }
<commentaire>"*"+"/"  { com--; if (!com) BEGIN(INITIAL); }
<commentaire>([^*/]*("*"+[^*/])*("/"+[^*/])*)*
<commentaire><<EOF>>    { printf("commentaire non terminé!\n"); exit(1); }
...
```

Correction de l'exercice 7 :

```
%{
#include <strings.h>
int nbdef = 0, i;
char *clef[100], *def[100];
%}
%x define1
%x define2
%%
^".def "                     { BEGIN(define1); }
[a-zA-Z]+                    { for(i=0;i<nbdef;i++)
                                 if (!strcmp(clef[i],yytext)) break;
                               printf("%s",(i<nbdef)?def[i] :yytext); }

<define1>[a-zA-Z]+           { clef[nbdef]=strdup(yytext); }
<define1>" "+                { BEGIN(define2); }
<define1>.                   { printf(".def doit etre suivi d'un mot");exit(1); }

<define2>.*                  { def[nbdef++]=strdup(yytext); }
<define2>\n                  { BEGIN(INITIAL);}
...
```

Correction de l'exercice 8 :

```
%{
YY_BUFFER_STATE inc[100];
int nn = 0;
%}
%x incl
%%
^".include "          { BEGIN(incl); }
<<EOF>>               { if (!nn) yyterminate();
                        else { yy_switch_to_buffer(inc[--nn]); BEGIN(incl); } }

<incl>[ ]+
<incl>[a-zA-Z0-9.]+   { inc[nn++]=YY_CURRENT_BUFFER;
                        yy_switch_to_buffer(
                          yy_create_buffer(fopen( yytext, "r" ),
                          YY_BUF_SIZE));
                        BEGIN(INITIAL); }
<inc>\n               { BEGIN(INITIAL); }
...
```