Answers to the exercises for chapter: *lex*

1.
```
%{ /* Declarations */

#include <stdlib.h>
#include <string.h>

#define STACKSIZE 100
  static int stack[STACKSIZE];
  static int sp; /* stack pointer is the first free location */
  static void push(int);
  static int pop(void);

%}

/* no lex definitions needed in this example */

%%

 /* inputs and actions */

[0-9]+           {push(atoi(yytext));}
"+"              {push(pop()+pop());}
"-"              {int r = pop(); push(r-pop());}
"*"              {push(pop()*pop());}
"/"              {int r = pop(); push(r/pop());}
"\n"             {if (sp>0) {
                      printf("Result: %d\n",pop());
                      if (sp>0) printf("Warning: stack not empty\n");
                      }
                 }
[ \t]            ;
%%

 /* main and auxiliary routines */
static void push(int x)
{
  if (++sp>=STACKSIZE) {printf("Stack overflow\n"); exit(1);}
  stack[sp] = x;
  return;
}
```

```
static int pop(void)
{
  if (sp<=0) {printf("Stack underflow\n"); exit(1);}
  return stack[sp--];
}

int main(void)
{
  sp = 0;
  yylex();
  return 0;
}
```

2. The characters would be misinterpreted as the end of the group or an indication of a range respectively, otherwise.

3.
```
^[^a]*\a
^.*]a
a.*$
a[^a]*$
```

4.
```
%{

%}

letter [a-zA-Z]
nonletspace [^a-zA-Z ]
ws [ \t]

%s N
%s S
%s M

%%


\\{letter}+ {printf("<cseq: %s>",yytext+1); BEGIN S;}
\\{nonletspace}       {printf("<csym: %s>",yytext+1); BEGIN M;}
\\[ ] {printf("<cspace>"); BEGIN S;}
\{ {printf("<{>"); BEGIN M;}
\} {printf("<}>"); BEGIN M;}
\%.*\n {printf("<comment>\n");}
<N>{ws}+  ;
<S>{ws}+  ;
<M>{ws}+ {printf("<space>"); BEGIN S;}
[^ \t\n] {ECHO; BEGIN M;}
<N>\n     ;
<M>\n     {printf("<sp>"); ECHO; BEGIN N;}
<S>\n     {ECHO; BEGIN N;}

%%

int main()
```

```
{
yylex();
return 0;
}
```