```
#######on page 4
     Symbolic Math Toolbox              Version 7.2     (R2017a)
#######replaced by
     Symbolic Math Toolbox              Version 8.2     (R2018b)
################################################################

#######on page 5
        This manual assumes that you are using Version 7.2
#######replaced by
        This manual assumes that you are using Version 8.2
################################################################

#######on page 6
```

Consider

```
    a = sym(6)
    b = sym('2*a')
    c = sym(2)*a
```

What happened? The result in b might not be what you wanted!
This suggests you should avoid putting expressions inside the
quotes and instead build them out of symbolic variables or symbolic
numbers.
For example, rather than writing f = sym('a * x + b'), it's
probably better to do

```
    syms a x b
    f = a*x + b
```

In fact it is discouraged to pass anything other than numbers and
variable names to sym. For example

```
  c = sym('sqrt(2)')
```

works but results in an unfriendly warning message from Matlab.

```
#######replaced by
```

Consider

```
    a = sym(6)
    b = sym(2)*a
    c = sym('2*a')
```

What happened? The result in b is correct, but c throws an error.
This is because Matlab does not want you to write expressions inside
quotes and pass them to sym.
For example, instead of f = sym('a * x + b'), you should write

```
    syms a x b
    f = a*x + b
```

If you really need to pass a string instead of a number or a
variable to sym, you can use the command str2sym. For example,
you can write

```
  str2sym('sqrt(2)')
```

However, passing expressions as strings may not always produce the desired result. For instance, compare the outputs of the following commands.

```
    a = sym(6)
    b = sym(2)*a
    c = str2sym('2*a')
```
##