

```
function Results = toxClassAlg(doseIndex, logDoseArray, log_trueLD50, log_sigma,
doseGroupSize)
% This function implements the toxic classification outlined in the OECD
% guidelines.
%
% Inputs: doseIndex - the integer index of the array logDoseArray we are
%          beginning with
%          logDoseArray - a double array of the log (base 10) dose values
%          log_trueLD50 - a double representing the log (base 10) of the
%                         reported "true" LD50 value for the particular chemical
%          log_sigma - a double value of the log (base 10) standard deviation of LD50
values
%          doseGroupSize - the integer number of animals in a dose group
%
% Outputs: Results - a structure containing information fields:
%          numberAnimalsDosed - # animals dosed in the algorithm
%          numberAnimalsDead - # animals that died during the
%                               algorithm run
%          category - a scalar (valued in 1-6) representing
%                      the GHS category the algorithm defined
%
% NOTE: Ensure that the units are the same for the 'logDoseArray',  

% 'log_trueLD50', and 'log_sigma' values.
```

% Define incremental parameters.

```
doseInc = 0; % represents whether the dose has been increased (value = 1) during the
study
```

```
doseDec = 0; % represents whether the dose has been decreased (value = 1) during the
study
```

```
numberDead = 0; % represents number of animals that die
```

```
numberDosed = 0; % represents number of animals that are dosed
```

% Define start dose.

```
log_dose = logDoseArray(doseIndex);
```

% Keep trying doses until we reach the lowest or highest dose possible.

```
while ( (doseIndex > 0) & (doseIndex <= length(logDoseArray)) )
```

```
numberDeadTemp = doseResponse(log_dose, log_trueLD50, log_sigma, doseGroupSize);
```

```
numberDead = numberDead + numberDeadTemp;
```

```
numberDosed = numberDosed + doseGroupSize;
```

```
if numberDeadTemp <= 1 % if first try at dose kills 0-1 animals
```

```
numberDeadTemp = doseResponse(log_dose, log_trueLD50, log_sigma, doseGroupSize);
```

```
numberDead = numberDead + numberDeadTemp;
```

```
numberDosed = numberDosed + doseGroupSize;
```

```
if numberDeadTemp <= 1 % if second try at dose kills 0-1 animals

    if ( (doseDec == 1) | (doseIndex == length(logDoseArray)) ) % if at the
highest dose OR the dose has been decreased

        switch doseIndex

            case 1
                category = 2;
            case 2
                category = 3;
            case 3
                category = 4;
            case 4
                if numberDeadTemp == 0 % if no animals died
                    category = 6;
                else % if 1 animal died
                    category = 5;
                end
            otherwise
                error('Unknown category')
            end
        break; % END THIS SIMULATION!

    else

        doseInc = 1;
        doseIndex = doseIndex + 1;
        log_dose = logDoseArray(doseIndex);

    end

end

if numberDeadTemp > 1 % if ANY (first or second) try at dose kills 2-3 animals

    if ( (doseInc == 1) | (doseIndex == 1) ) % if at the lowest dose OR the dose has
been increased

        switch doseIndex

            case 1
                category = 1;
            case 2
                category = 2;
            case 3
```

```
    category = 3;
case 4
    category = 4;
otherwise
    error('Unknown category')

end
break; % END THIS SIMULATION!

else % if dose has been decreasing AND not at the lowest dose

    doseDec = 1;
    doseIndex = doseIndex - 1;
    log_dose = logDoseArray(doseIndex);

end

end

% Compose output structure.
Results.numberAnimalsDosed = numberDosed;
Results.numberAnimalsDead = numberDead;
Results.category = category;
```