```
if not modules then modules = { } end modules ['deadstone'] = {
  version = 0.100,
  comment = "Dead stone calculator for go",
  author = "Wolfgang Schuster",
  copyright = "Wolfgang Schuster",
}
```

1 This is the *Lua* version of my deadstone calculator. It is a nearly one to one copy of my original idea of my method, I developed to find the dead stones on the goban.

```
laststone = 1
```

2 The **function** clearboard is called at the begin of a new board to initialize all fields with the value 0 and the margins with the value 3. I need this values for the later calculations of the field states.

```
function clearboard()
 field = {}
 for y=0,tex.count.boardsize+1 do
  field[y] = {}
  for x=0,tex.count.boardsize+1 do
   if x==0 then
    field[y][x] = { 3, 0 }
   elseif x==tex.count.boardsize+1 then
    field[y][x] = { 3, 0 }
   elseif y==0 then
    field[y][x] = { 3, 0 }
   elseif y==tex.count.boardsize+1 then
    field[y][x] = { 3, 0 }
   else
    field[y][x] = { 0, 0 }
   end
  end
 end
end
```

3 To decide wheter I handle in the calculation a white or a black stone the state for stone and enemy depends of the current color. The value for marked depends also on the current stone color to reset already living stones in the last step back to the normal color. This could have been done also by checking the current stone color but two different values work without this test. The values for wall, error (should never happen) and dead is the same for both, black and white stones. There is no need to make a difference between them.

```
function blackstone()
 stone=1
```

```
 enemy=2
 wall=3
 marked=4
 error=6
 dead=7
end

function whitestone()
 stone=2
 enemy=1
 wall=3
 marked=5
 error=6
 dead=7
end
```

4 The **function** doprocesstones is used to call all subfunctions. It is called twice, once for the black stones and a second time for the white stones.

```
function doprocesstones()
 markstones()
 deadstones()
 -- for n=1,tex.count.deadcount
 -- checkstones()
 -- end
 checkstones()
 checkstones()
 revertstones()
end
```

5 The order in which the last **function** doporocesstones ist called depends on the current placed stones. This mean if the last stone was a black one, we look at the begin for dead white stone and reverse if the last stone was a white one. We could check the current stone color with the variable laststone and test for the numeric value.

```
function processtones()
 -- we placed a black stone
 if laststone==1 then
  processwhitetones()
  processblackstones()
 -- we placed a white stone
 elseif laststone==2 then
  processblackstones()
  processwhitetones()
```

```
 end
end

function processblackstones()
 blackstone()
 doprocesstones()
end

function processwhitetones()
 whitestone()
 doprocesstones()
end
```

6 The **function** markstones mark the stones if they have the right stones on their sides or let them keep untouched.

```
function markstones()
 for y=1,tex.count.boardsize do
  for x=1,tex.count.boardsize do
   if field[y][x][1]==stone then
    if (field[y][x-1][1]==wall or field[y][x-1][1]==enemy or field[y][x-1][1]==marked)
     and (field[y][x+1][1]==wall or field[y][x+1][1]==enemy or field[y][x+1][1]==stone)
     and (field[y-1][x][1]==wall or field[y-1][x][1]==enemy or field[y-1][x][1]==marked)
     and (field[y+1][x][1]==wall or field[y+1][x][1]==enemy or field[y+1][x][1]==stone)
    then
     field[y][x] = { marked, field[y][x][2] }
    end
   end
  end
 end
end
```

7 The **function** deadstones set stones with the value marked to dead if the conditions in the function are true.

```
function deadstones()
 for y=tex.count.boardsize,1,-1 do
  for x=tex.count.boardsize,1,-1 do
   if field[y][x][1]==marked then
    if (field[y][x-1][1]==wall or field[y][x-1][1]==enemy or field[y][x-1][1]==marked)
     and (field[y][x+1][1]==enemy or field[y][x+1][1]==wall or field[y][x+1][1]==dead)
     and (field[y+1][x][1]==enemy or field[y+1][x][1]==wall or field[y+1][x][1]==dead)
    then
     field[y][x] = { dead, field[y][x][2] }
```

```
      end
     end
    end
   end
  end
```

8 Because the **function** deadstones can sometimes set already living stones to dead. To prevent this in the final result this function looks through all stones with a loop in reverse direction and reset the values to their original value if the stone is not dead and should remain on the board. The function is currently called twice within processtones but this can be changed with the counter deadcount.

```
function checkstones()
 for y=1,tex.count.boardsize do
  for x=1,tex.count.boardsize do
   if field[y][x][1]==marked then
    field[y][x] = { stone, field[y][x][2] }
   elseif field[y][x][1]==dead then
    if (field[y][x-1][1]==dead or field[y][x-1][1]==enemy or field[y][x-1][1]==wall)
     and (field[y][x+1][1]==dead or field[y][x+1][1]==enemy or field[y][x+1][1]==wall)
     and (field[y-1][x][1]==dead or field[y-1][x][1]==enemy or field[y-1][x][1]==wall)
     and (field[y+1][x][1]==dead or field[y+1][x][1]==enemy or field[y+1][x][1]==wall)
    then
     field[y][x] = { dead, field[y][x][2] }
    else
     field[y][x] = { stone, field[y][x][2] }
    end
   end
  end
 end
end
```

9 The last thing to do after all dead stones are found on the board is to remove them and to reset all other stones which are still in a marked state or we will get wrong input for the next move.

```
function revertstones()
 for y=1,tex.count.boardsize do
  for x=1,tex.count.boardsize do
   if field[y][x][1]==marked then
    field[y][x] = { stone, field[y][x][2] }
   elseif field[y][x][1]==dead then
    appendtodeadstonelist(y,x,stone,field[y][x][2]) -- append to deadstonelist
    field[y][x] = { 0, 0 } -- before reseting the field
```

```
    end
   end
  end
end
```

To use the dead stones we found in this run on the board for later use they are saved in two commalist, one for white stones and another one for the black ones. They can be used by other macros, the only thing here is done is to append the removed stones in the current run to the already romved stones in one of the former calculations.

```
function appendtodeadstonelist(row,column,color,count)
 if color==1 then
  tex.print("\\expanded{\\appendtocommalist{{"
   .. row .. ":" .. column .. ":1:" .. count .. "}}}\\deadblackstones")
 elseif color==2 then
  tex.print("\\expanded{\\appendtocommalist{{"
   .. row .. ":" .. column .. ":2:" .. count .. "}}}\\deadwhitestones")
 end
end
```

Functions

Variables