```lua
-- standard load and  start functions
-- first line of code should be a help string (e.g. see tiny.lua)
-- last line  of code should call this code, pass in table of actions
-- e.g
--        the(go)

-------------------------------------------------------------------------------
-- at load time, remember the current globals
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
-- after start  time, complain if code has created  rogue globals
local function rogues()
  for k,v in pairs(_ENV) do if not b4[k] then print("?:",k,type(v)) end end end

-------------------------------------------------------------------------------
-- Misc support functions. Nothing very exciting.
-- table keys, in sorted order
local function keys(t,u)
  u={}; for k,_ in pairs(t) do u[1+#u]=k end;  table.sort(u); return u end

-- pretty colors, n={31,32},={red,green}
local function color(n,s) return string.format("\27[1m\27[%sm%s\27[0m",n,s) end

-- shallow copy of a list
local function copy(t,  u)
     u={}; for k,v in pairs(t) do u[k]=v end ; return u end

-- coerce strings to nums
function string2value(x)
  if x=="false" then return false  end
  if x=="true"  then return true   end
  return tonumber(x) or x end

-------------------------------------------------------------------------------
-- More interesting stuff to handle load and start
local help="" -- place to store the help test

-- if "-flag" matches to something on the command line, then update flag's value
local function updateFlagFromCommandLine(flag,x)
  for n,word in ipairs(arg) do
    if flag:match("^"..word:sub(2)..".*") then
       -- flip boolean defaults
       x= (x=="false" and "true") or (x=="true" and "false") or arg[n+1] end end
  return string2value(x) end

-- all the start-up action:
-- [1] keep a copy of the options as "defaults"
-- [2] maybe just show the  help text
-- [3] maybe run an  action in verbose mode (show stackdump; halt on error)
-- [4] before actions, reset options to detaults
-- [5] before actions, reset random number seed
-- [6] maybe  run an  action in fast mode (no stackdumps; no halts one errors)
-- [7] for fast mode, count the number of failures
-- [8] return to the operating system the count of failures
-- [9] lint the code (right now, we just print rogue globals)
local function what2doAtLastLine(options, actions)
  local fails, defaults = 0, copy(options)             -- [1]
  if options.h     then return print(help) end         -- [2]
  if options.debug then actions[ options.debug ]() end -- [3]
  local todos = options.todo =="all" and keys(actions) or {options.todo}
  for _,todo in pairs(todos) do
    if type(actions[todo]) ~= "function" then return print("NOFUN:",todo) end
    for k,v in pairs(defaults) do options[k]=v end    -- [4]
    options.seed = options.seed or 10019              -- [5]
    local ok,msg = pcall( actions[todo] )             -- [6]
    if not ok then print(color(31,"FAIL ")..todo,msg) -- [6]
                   fails=fails+1 end                  -- [6] [7]
  end
  rogues()            -- [9]
  os.exit(fails) end -- [8]

local function what2doAtFirstLine(txt)
  local options={}
  help = txt
  txt:gsub("^.*OPTIONS:",""):gsub("\n%s*-([^%s]+)[^\n]*%s([^%s]+)",
           function(flag,x) options[flag] = updateFlagFromCommandLine(flag,x) end)
  return setmetatable(options,{__call=what2doAtLastLine}) end

return what2doAtFirstLine
```