Assignment: Problem solving exercises

Individual Assessment worth 15%

Title: Cumulative modification of code

Related outcomes from the unit outline:

1. Program a simple mobile application.
2. Deploy a mobile application successfully.

Due date: 22/09/2017, 11:59pm, WST.

Suggested Length: None specified.

Submission instructions:

To be submitted electronically in Blackboard

Plagiarism:

Please ensure that you have read and understood the information on academic misconduct provided on Blackboard.

Referencing:

Not required

Background:

Modification of existing code allows students to gain insight about another developer's experience and is a useful way to develop problem-solving skills. Listing 1 contains part of a noughts and crosses app. The developer has had some trouble with event handling and has asked another developer to help. The result is listing 2.

Task:

- Complete the Corona app in Lua to allow the placement of noughts (O) and crosses (X) within the grid provided.
- Allow a human to play vs. "the computer"
- Allow a choice as to who goes first
- Keep track of the moves made and be able to show the winner
- Use a consistent coding standard, either open source or self-defined
- Provide a Word document containing a record of the black box and unit tests you have completed to verify that your app works. No other document formats accepted.
- Provide a zip file containing your assignment. No other compression formats accepted. The file should contain your project, including code and any other ancillary files required to run your code in the Corona simulator.
- Separately (not in the zip file), provide the MD5 hash value of your zip file. Submissions without a hash value will not be accepted or assessed.

Important: You do **not** have to implement any logic for a winning strategy.

```
d = display
w20 = d.contentWidth * .2
h20 = d.contentHeight * .2
w40 = d.contentWidth * .4
h40 = d.contentHeight * .4
w60 = d.contentWidth * .6
h60 = d.contentHeight * .6
w80 = d.contentWidth * .8
h80 = d.contentHeight * .8


----DRAW LINES FOR BOARD
local lline = d.newLine(w40,h20,w40,h80 )
lline.strokeWidth = 5

local rline = d.newLine(w60,h20,w60,h80 )
rline.strokeWidth = 5

local bline = d.newLine(w20,h40,w80,h40 )
bline.strokeWidth = 5

local tline = d.newLine(w20,h60,w80,h60 )
tline.strokeWidth = 5


--PLACE BOARD COMPARTMENT DIMENSIONS IN TABLE
board ={

{"tl", 1, w20, h40, w40, h20,0},
{"tm",2, w40,h40,w60,h20,0},
{"tr",3, w60,h40,w80,h20,0},

{"ml", 4, w20, h60, w40, h40,0},
{"mm",5, w40,h60,w60,h40,0},
{"mr",6, w60,h60,w80,h40,0},

{"bl", 7, w20, h80, w40, h60,0},
{"bm",8, w40,h80,w60,h60,0},
{"br",9, w60,h80,w80,h60,0}
}
--

--FILL COMPARTMENT W/ COLOUR WHEN TOUCHED
local function fill (event)
if event.phase == "began" then
      tap = 0

      for t = 1, 9 do
      if event.x > board[t][3] and event.x < board [t][5] then
            if event.y < board[t][4] and event.y > board[t][6]
            then
```

```
            r = d.newRect(board[t][3],board [t][6],w20,h20)
                  r:setFillColor(1,1,0)
                  r.anchorX=0
                  r.anchorY=0
                  end
            end
      end
end

end
Runtime:addEventListener("touch", fill)
```

Listing 1: Sample XOX Code.

```
local EMPTY, X, O = 0, 1, 2
local whichTurn = X -- X is starting game

...

--FILL COMPARTMENT W/ COLOUR WHEN TOUCHED
local function fill (event)
if event.phase == "began" then
      for t = 1, 9 do
            if event.x > board[t][3] and event.x < board [t][5]
      then
            if event.y < board[t][4] and event.y > board[t][6]
      then
            if board[t][7] == EMPTY then
                  board[t][7] = whichTurn
                  whichTurn = whichTurn == X and O or X
                  end
            end
      end
end
end

end
Runtime:addEventListener("touch", fill)
```

Listing 2: Alternate Event Handler Code.