FINAL HOMEWORK MAY 1 2016

You had an assignment to press the left button once, turn the purple LED ON. Press it again, turn the purple LED OFF. And so on…. Each time the left button is pressed, the purple led should "**TOGGLE**."

The code below was suggested in class but **DOES NOT WORK:**

```
if leftbutton() and (x=1) then
   PurpleLed (1)
   x=0
end if
if leftbutton () and (x=0) then
   PurpleLed (0)
   x = 1
end if
```

The reason this does not work is due to the fact that when the leftbutton() command is executed, it returns TRUE if the left button was pressed and returns FALSE if it was NOT pressed. BUT IT ONLY RETURNS TRUE **ONE TIME** AFTER THE BUTTON IS PRESSED. (This is a good thing.)

So, let's say that x happens to equal zero (and the purple led is ON) and the program is running around and around in the infinite loop.

And you **press the button just before the program hits the FIRST IF..THEN block of code:**

```
If LeftButton() and (x=1) then
   PurpleLed (1)
end if
```

When this first IF… statement gets executed, the LeftButton() will be equal to TRUE, since it was just pressed. But Since x happens to equal zero at this time, the PurpleLed(1) statement will NOT be executed, **which is as it should be.**

But the problem is that now, the LeftButton() command has "shot its one shot" of being TRUE and will now be FALSE when the SECOND IF..THEN block of code gets executed:

```
If LeftButton() and (x=0) then
   PurpleLed (0)
end if
```

So, although x is equal to zero and we are hoping that the LeftButton() being pressed will be TRUE, which will allow PurpleLed(0) to execute and turn OFF the purple led, **THIS WON'T HAPPEN** because unfortunately, now, LeftButton() will be FALSE **(since it "shot its one shot" of being TRUE in the FIRST IF..THEN block of code.)**

SO THE CODE ABOVE WILL NOT WORK RELIABLY.

The way to do it is to use only ONE test that uses LeftButton() like so:

THIS WORKS:

```
if leftbutton() then
   if x=1 then
      purpleled(1)
      x=0
   else
      PurpleLed(0)
      x=1
   end if
end if
```

Or, you can do it this way, which uses 1 less line of code:

```
if leftbutton() then
   if x=1 then
      x=0
   else
      x=1
   end if
   PurpleLed (x)
end if
```

But the SUPREME WAY to toggle the purple LED would be to DIMENSION x as an INTEGER, and change your PurpleLed subroutine a bit. In the Infinite Loop, the code would be:

```
if leftbutton() then
   x = x * -1     'here we are using an INTEEGER TYPE Variable
   PurpleLed(x)   'here the PurpleLed subroutine was changed slightly - see it
end if
```

Above, every time the LeftButton is pressed, x will become x multiplied by minus one - - so x will change between plus one, then minus one, then plus one, then minus one …. and so on…

Below shows most of the code for this toggle:

```
dim x as integer
'  [Dimension all variables used in this area]

sub procedure PurpleLed (dim challenge as integer)
      'challenge can be +1 to turn on, anything else to turn off
      if challenge=1 then
         rgbled (1,0,1)
      else
         rgbled (0,0,0)
      end if
end sub

' This program [brief description here]
'-------------------------------------------------------------------------
'
'--------------------------M A I N ----------------------------------------
'
'-------------------------------------------------------------------------

'  Main program - This group of lines is part of the SCIO "SKELETON":
main: org (4096)  'They set up the Software Environment.
      orgall (4096)
      setup ()

      'USER PROGRAMMING AREA starts below this line:

' [Initialize variables in this area]
       x=1

    InfLoop:

    '[Put the main code inside this "infinite loop" area]
        if leftbutton() then
           x = x * -1     'here we are using an INTEEGER TYPE Variable
           PurpleLed(x)   'here the PurpleLed subroutine was changed slightly - see it
        end if

      goto InfLoop  'This statement causes the program to go back to the
                    'LABEL called "InfLoop".
```

HOMEWORK ASSIGNMENT:

Be able to show a reliable way of pressing the **right button** once to turn on a **YELLOW RGB LED**. Then press it again to turn the YELLOW LED OFF, and so on. Here, you program uses the RIGHT BUTTON to **TOGGLE** the state of the YELLOW LED.

TO THINK ABOUT, and get EXTRA HOMEWORK CREDIT:

When your hand moves over a light sensor, the light sensor value falls below 90, as we have seen in class. So you might think it is easy to write a program that will TOGGLE THE STATE OF THE YELLOW LED like requested above, but do it when you WAVE YOUR HAND OVER THE LIGHT SENSOR.

But this is not as easy as you might think. What would happen if you tried to do this as shown below ?:

```
If LightSensor(1) < 90 then
  if x=1 then
     YellowLed(1)
     x = 0
  else
     YellowLed(0)
     x = 1
   end if
end if
```

The problem here is that the Infinite Loop gets executed hundreds of thousands of times every second. So if your hand STAYED OVER THE LIGHT SENSOR LONGER THAN A FEW MICROSECONDS (a microsecond is a millionth of a second), the LightSensor(1) would STILL BE under 90 the next time around, and the light would probably turn ON and OFF thousands of times very quickly before the shadow of your hand moved off of the light sensor. The state of the Yellow LED would be left INDETERMINATE - - it might be ON, or it might be OFF when your hand gets out of the way!

So how would you fix this problem? Again, you will need to think out of the box a bit. But with JUST ONE MORE STATEMENT you can fix this and make it work, for EXTRA CREDIT.