COMPUTER SCIENCE 110

**INTRODUCTION TO COMPUTER SCIENCE**

**HOMEWORK 2,** Spring 2018 – Multimedia Python

**Assigned**: Wednesday, February 7 **Due**: **by beginning of class** Monday, February 19

BD14710_

**Part I (60 points) Representing Text and Colors; Boolean logic, expressions, gates, circuits** (**show your work; it’s all right to hand-write the work, but be neat (or type if you can’t be neat): take pride in the product of your efforts!). Put your answers on another sheet of paper, not this one**:

1. (5 pts) How would the following string of characters be represented using run-length encoding? What is the compression ratio?

OOOOOOOO NNooooo!!! New Enggggland lllooooooosssssssttt!!!!!

\*O8 NN\*o5!!! New En\*g4land lll\*o7\*s7ttt\*!5

compression ration: 42/60 = .7

1. (4 pts) What do the following ASCII character codes spell?

01000011 00110000 01101101 01110000 01110101 01110100 01000101 01010010 00100000 00100100 01100011 01101001 01100101 01101110 01100011 01100101 00100001

C0mputER $cience!

1. Huffman codes: Use the Huffman code at the top right of page 94 of the CSI textbook to answer the following (6 pts):
   1. Write the Huffman code for the word “FROSTED”

101001 1011 10010 1000 010 11 101011

* 1. Decipher the Huffman code: 1001110100010000101100101011

INSTEAD

* 1. Decipher the Huffman code: 100000100111010000101000

SAINTS

1. (6 points) Use Microsoft Word (Font Colors | More Colors | Custom) or the Internet to find the RGB, or 6 **hex-digit** values – approximately – for the following colors:
2. Pink – something close to #ffb6c1 (255,182,193)
3. Purple – something close to #800080 (128,0,128)
4. Aqua-- something close to #00FFFF (0, 255, 255)

The colors will be approximate, but MUST be in the form of 6 **hex-digit values** (two for each of RGB).

1. (8 pts.) Consider the following two Boolean expressions:

A OR (B OR C)’

A’ OR (B’ OR C’)

Are they equivalent? Prove it!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **B or C** | **(B or C)’** | **A or (B or C)’** |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |

No, they are not equivalent. One way to prove this is to draw the truth tables for each one and notice that the output columns of the truth tables differ:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A’** | **B’** | **C’** | **B’ or C’** | **A’ or (B’ or C’)** |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

1. (6 points) For the following, assume that A = true, B = false, and C = true.  Evaluate each expression below (i.e., state whether it is TRUE or FALSE), **showing your work**:
2. ( A AND B’) OR ( C‘)

(T and F’) or (T’)

(T and T) or F

T or F

T

1. ( A OR B OR C) AND ( A’ AND B’ AND C’ ) AND ( A OR B’ OR C’ )

(T or F or T) and (T’ and F’ and T’) and (T or F’ or T’)

(T) and (F and T and F) and (T or T or F)

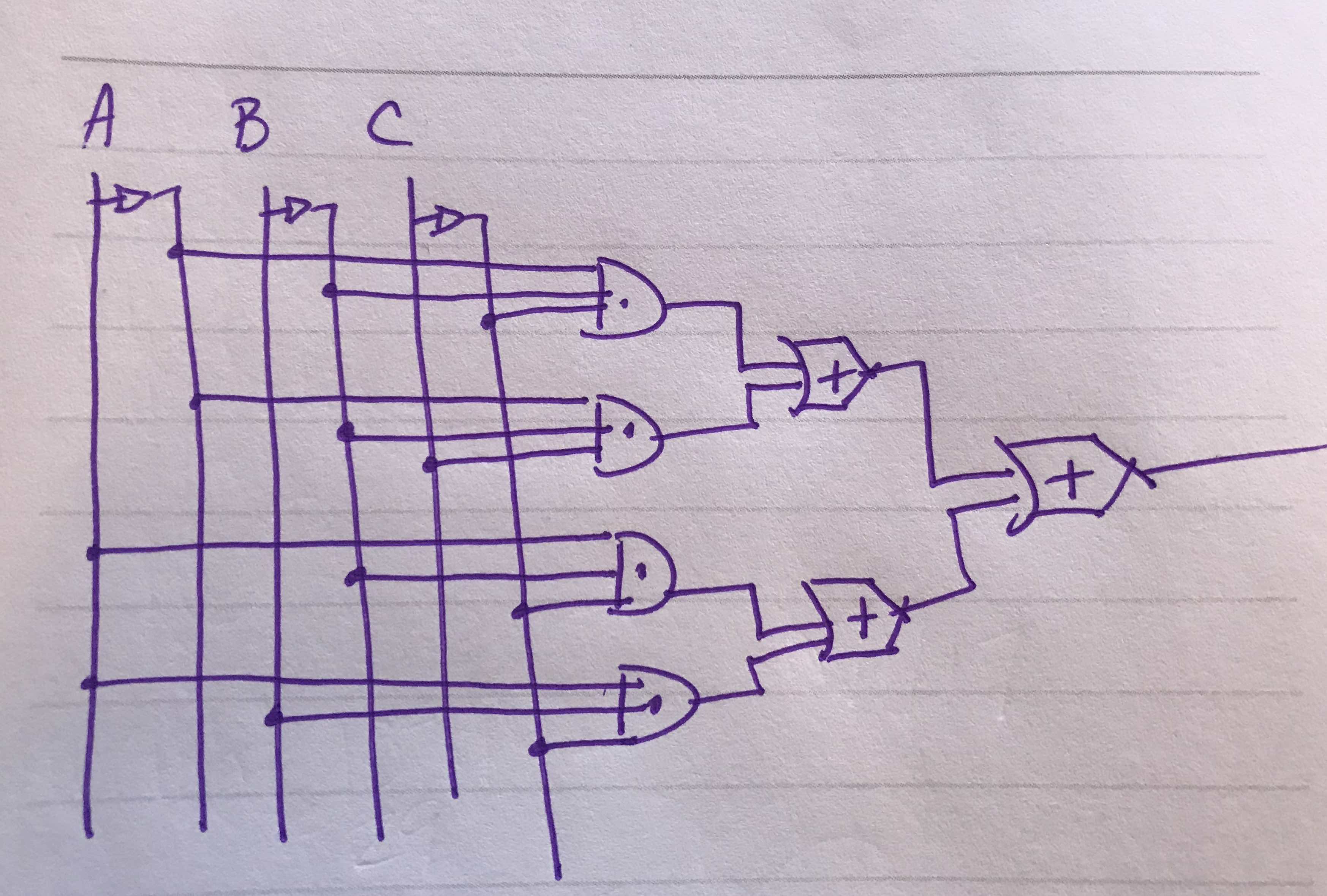
T and F and T

F

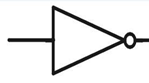
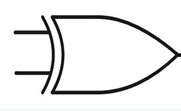
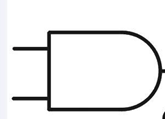
1. (16 Points) Design a circuit using the Sums-of-Products algorithm that implements the following truth table. For full credit, you must show the results from each step of the algorithm, including the final equivalent Boolean expression for this truth table.

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **Output** |
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **0** |
| 0 | 1 | 1 | **0** |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **0** |
| 1 | 1 | 0 | **1** |
| 1 | 1 | 1 | **0** |

A’B’C’ + A’B’C + AB’C’ + ABC’

****

1. (9 points; 3 each)
2. Draw a circuit diagram corresponding to the following Boolean expression: AB’ + (B + C)’
3. Show the behavior of the following circuit with a truth table



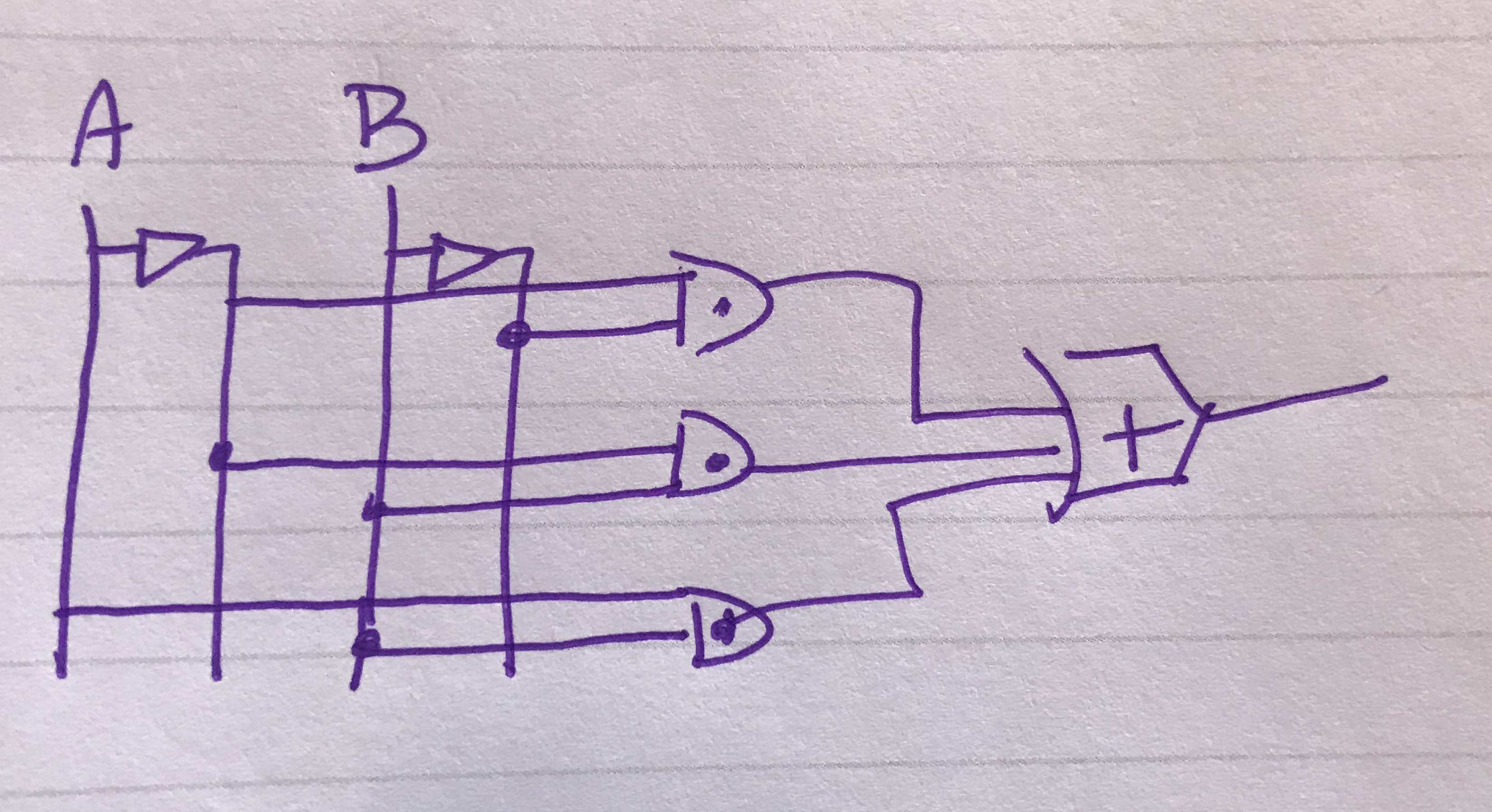
**A**

**B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **A’** | **A and B** | **A’ xor (A and B)** |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

1. Draw the circuit from Question H(2) using only AND, OR, and NOT gates (no XOR or other types of gates).

A’B’ + A’B + AB



BD14710_

**Part III (40 points) Python Programming**

Your Python code must follow these guidelines:

* *Always put your name, date, program name, and a general comment describing what your program does at the top of your code. This should be a general statement about how your program behaves.*
* *Variable names should help describe the purpose of a variable.*
* *Function names should help describe the purpose of a function*
* *Every function should be preceded by a comment describing the function’s purpose, input, and output (if any).*
* *Remove any of our comments that we put there to help you with your program.*

Be aware that if you don’t follow these guidelines, you will lose points, **even if your program runs perfectly**.

Write a function named **createCollage()** to create a collage of the same image at least four times, then mirror the collage. **Your function must be saved in a file called “hw2.py” in your csis110/hw/hw2 folder.** You can use any image you want, either from MediaSources or something you find on the web or a picture of your dog, or whatever. You should create a blank canvas of the proper size (as long as it fits on a normal monitor) using the Python function **makeEmptyPicture**. One of the four images in your collage must be the original, unaltered image. The others must be modified forms of the original image. You can scale, crop, rotate, create a negative, shift or alter colors, make it darker/lighter, or virtually anything else you want. Be creative and have fun with it!!

After making this image, mirror it (you will now have 8 “copies” of the picture). You can do this vertically or horizontally (or some other way), in any direction – just make sure that the original 4 images are still visible after mirroring. The **createCollage()** function you write must make all of this happen – all of the above effects and compositing must occur from the single function **createCollage()**. Of course, I expect you will want to write and use additional functions to make this easier, but someone (like me!) testing your program must be able to simply call **setMediaPath()**, load your program, then execute **createCollage()** to see your collage generated and displayed.

**NOTE**: A perfectly working program as described above will earn 35 points. To earn the remaining 5 points, you must use your own creativity to go above and beyond those minimal requirements. Examples include additional manipulations of the original picture, unique mirroring techniques, original or unique manipulations, etc. Go nuts!

**Tips**:

1. Start with the working program that we give you on Blackboard. Get that to work, then change it to use a different picture, add more manipulations, etc.
2. Build on what you’ve learned! There’s actually not much that’s new here, mostly putting together things you’ve already done. If you don’t have your book beside you and your code from the labs in front of you when you’re doing this, then you are going to waste a lot of time.
3. To get inspiration, check out some of the collages at <http://home.cc.gatech.edu/gacomputes/46>. Not all of these satisfy the assignment, and some are more ambitious than you have to be, but feel free to be inspired by them! I will also show you some collages from previous semesters for inspiration.

**TURNING IT IN:**

1. Turn in a printout of your code (the hw2.py file) with the rest of your homework.
2. Put a copy of your hw2.py file and your original image (not the collage) in your csis110/hw/hw2 folder. THIS IS REQUIRED IN ORDER FOR ME TO GRADE IT AND FOR YOU TO EARN FULL CREDIT.

# CSIS110, HW2, MaryAnne Egan

# Solution code and template

# Program creates a collage of 4 copies of the "kass.jpg" picture,

# the original, one negative, one posterized and one that is a

# mirror image of it. It then mirrors the whole

# collage, without losing any of the original 4 versions.

# Transforms "picture" into its negative (code from the book, Program 16)

def negative(picture):

for pixel in getPixels(picture):

r = getRed(pixel)

g = getGreen(pixel)

b = getBlue(pixel)

nucolor = makeColor(255-r, 255-g, 255-b)

setColor(pixel, nucolor)

# Mirrors an image horizontally (from book, Program 21)

def mirrorHorizontal(source):

mirrorPoint = int(getHeight(source)/2)

for yOffset in range(0, mirrorPoint):

for x in range(1,getWidth(source)):

pbottom = getPixel(source, x, yOffset+mirrorPoint)

ptop = getPixel(source, x, mirrorPoint-yOffset)

setColor(pbottom, getColor(ptop))

# Flip the image vertically

def flip(pic):

width = getWidth(pic)

height = getHeight(pic)

newPic = makeEmptyPicture(width, height)

for y in range(0, height):

for x in range(1, width):

pOld = getPixel(pic, x, y)

pNew = getPixel(newPic, width-x, y)

setColor(pNew, getColor(pOld))

copyInto(newPic, pic, 0, 0)

# Posterize the image

def posterize(picture):

for p in getPixels(picture):

red = getRed(p)

green = getGreen(p)

blue = getBlue(p)

if (red<64):

setRed(p, 31)

elif (red>63 and red<128):

setRed(p, 95)

elif (red>127 and red<192):

setRed(p, 150)

elif (red>191 and red<256):

setRed(p, 223)

if (green<64):

setGreen(p, 31)

elif (green>63 and green<128):

setGreen(p, 95)

elif (green>127 and green<192):

setGreen(p, 150)

elif (green>191 and green<256):

setGreen(p, 223)

if (blue<64):

setBlue(p, 31)

elif (blue>63 and blue<128):

setBlue(p, 95)

elif (blue>127 and blue<192):

setBlue(p, 150)

elif (blue>191 and blue<256):

setBlue(p, 223)

# Determine luminance of a given pixel

def luminance(pixel):

r = getRed(pixel)

g = getGreen(pixel)

b = getBlue(pixel)

return (r+g+b)/3

#turn edges into black lines and the rest of image white

def edgeDetect(pic):

for px in getPixels(pic):

x = getX(px)

y = getY(px)

if y<getHeight(pic)-1 and x<getWidth(pic)-1:

botRt = getPixel(pic, x+1, y+1)

thisLum = luminance(px)

brLum = luminance(botRt)

if abs(brLum - thisLum)>10:

setColor(px, black)

else:

setColor(px, white)

# Create a collage of 4 versions of "kass.jpg", then mirror

# horizontally. (adapted from book Program 31)

def createCollage():

# Find original picture and copy into empty canvas

orig = makePicture(getMediaPath("kass.jpg"))

widthOrig = getWidth(orig)

heightOrig = getHeight(orig)

canvas = makeEmptyPicture(widthOrig\*4, heightOrig\*2)

copyInto (orig, canvas, 0, 0)

# Negative original and insert into canvas

negative (orig)

copyInto (orig, canvas, widthOrig, 0)

# Get the original picture again, then vertically flip

# it, then posterize it, then insert into canvas

orig = makePicture(getMediaPath("kass.jpg"))

flip(orig)

posterize(orig)

copyInto (orig, canvas, widthOrig\*2, 0)

# Get the original picture again, then vertically flip

# it, then run edge detector, then insert into canvas

orig = makePicture(getMediaPath("kass.jpg"))

flip(orig)

edgeDetect(orig)

copyInto (orig, canvas, widthOrig\*3, 0)

# Mirror across horizontal axis

mirrorHorizontal (canvas)

# Wrap up

show(canvas)

writePictureTo(canvas, "kassCollage.jpg")