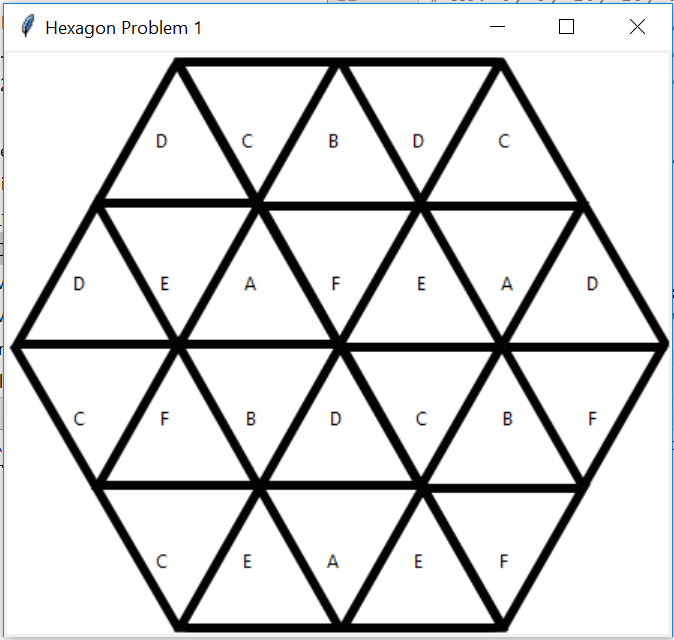
Note: sorry Dr. Gabor, I kinda read the assignment sheet after doing the assignment, and this would’ve been a bit nicer looking had I read it…my bad.

Also, I got no solution for question 2, although I did get one if there were 8 possible labels, but that wasn’t the question.

Question 1 Code:

From tkinter import \*

startPzl = **'........................'** *# each for an empty triangle*setOfChoices = {**'A'**, **'B'**, **'C'**, **'D'**, **'E'**, **'F'**} *# possible labels***def** copies(tpl): *# checks for copies within subhexagons* seen = set()  
 **for** label **in** tpl:  
 **if** label != **'.'**:  
 **if** label **in** seen:  
 **return True** *# true if it finds a copied label* **else**:  
 seen.add(label)  
 **return False** *# otherwise it's still valid***def** isInvalid(hexPzl): *# returns true if pzl is invalid, do better later maybe* hexTL = (hexPzl[0], hexPzl[1], hexPzl[2], hexPzl[6], hexPzl[7], hexPzl[8])  
 hexTR = (hexPzl[2], hexPzl[3], hexPzl[4], hexPzl[8], hexPzl[9], hexPzl[10])  
 hexML = (hexPzl[5], hexPzl[6], hexPzl[7], hexPzl[12], hexPzl[13], hexPzl[14])  
 hexMC = (hexPzl[7], hexPzl[8], hexPzl[9], hexPzl[14], hexPzl[15], hexPzl[16])  
 hexMR = (hexPzl[9], hexPzl[10], hexPzl[11], hexPzl[16], hexPzl[17], hexPzl[18])  
 hexBL = (hexPzl[13], hexPzl[14], hexPzl[15], hexPzl[19], hexPzl[20], hexPzl[21])  
 hexBR = (hexPzl[15], hexPzl[16], hexPzl[17], hexPzl[21], hexPzl[22], hexPzl[23])  
  
 **if** copies(hexTL) **or** copies(hexTR) **or** copies(hexML) \  
 **or** copies(hexMC) **or** copies(hexMR) **or** copies(hexBL) **or** copies(hexBR):  
 **return True** *# true if the overall puzzle contains a subhex with copies, meaning its invalid***def** bruteForce(pzl): *# find a solution through bruteForcing recursively* **if** isInvalid(pzl): *# throw it out if it's invalid* **return ''  
 if '.' not in** pzl: *# if there are no empty spaces and it's not thrown out, it's solved* **return** pzl  
  
 nextOpenIndex = pzl.find(**'.'**)  
 **for** choice **in** setOfChoices:  
 *# make a new pzl with the next empty choice filled with a possible label* subPzl = pzl[0:nextOpenIndex] + choice + pzl[nextOpenIndex + 1:]  
 result = bruteForce(subPzl)  
 **if** result != **''**: *# if it didn't fail, return it!* **return** result  
  
 **return ''** *# failure*solution = bruteForce(startPzl)  
print(solution)

*# displaying makes it easy to check visually*window = Tk()  
window.title(**'Hexagon Problem 1'**)  
hexPhoto = PhotoImage(file=**'hexagon.gif'**)  
pW, pH = hexPhoto.width(), hexPhoto.height()  
window.geometry(**'{}x{}'**.format(pW, pH))  
canvas = Canvas(window, width=pW, height=pH)  
canvas.pack()  
canvas.create\_image(pW/2, pH/2, image=hexPhoto)  
  
slnR1, slnR2 = solution[0:5], solution[5:12],  
slnR3, slnR4 = solution[12:19], solution[19:24]  
  
**for** index, label **in** enumerate(slnR1):  
 canvas.create\_text(105+57\*index, 60, text=label)  
**for** index, label **in** enumerate(slnR2):  
 canvas.create\_text(50+57\*index, 155, text=label)  
**for** index, label **in** enumerate(slnR3):  
 canvas.create\_text(50+57\*index, 245, text=label)  
**for** index, label **in** enumerate(slnR4):  
 canvas.create\_text(105+57\*index, 340, text=label)  
  
window.mainloop()

Question 2 Code: (Note: Not all of it because it’s exactly the same as the Q1 code. The only change is in isInvalid which I copied below, and an additional label ‘G’ for the seventh.)

**def** isInvalid(hexPzl): *# returns true if pzl is invalid, do better later maybe* hexTL = (hexPzl[0], hexPzl[1], hexPzl[2], hexPzl[6], hexPzl[7], hexPzl[8])  
 hexTR = (hexPzl[2], hexPzl[3], hexPzl[4], hexPzl[8], hexPzl[9], hexPzl[10])  
 hexML = (hexPzl[5], hexPzl[6], hexPzl[7], hexPzl[12], hexPzl[13], hexPzl[14])  
 hexMC = (hexPzl[7], hexPzl[8], hexPzl[9], hexPzl[14], hexPzl[15], hexPzl[16])  
 hexMR = (hexPzl[9], hexPzl[10], hexPzl[11], hexPzl[16], hexPzl[17], hexPzl[18])  
 hexBL = (hexPzl[13], hexPzl[14], hexPzl[15], hexPzl[19], hexPzl[20], hexPzl[21])  
 hexBR = (hexPzl[15], hexPzl[16], hexPzl[17], hexPzl[21], hexPzl[22], hexPzl[23])  
  
*# adding in rows and diagonals for Q2:*

H1 = (hexPzl[0], hexPzl[1], hexPzl[2], hexPzl[3], hexPzl[4])  
H2 = (hexPzl[5], hexPzl[6], hexPzl[7], hexPzl[8], hexPzl[9], hexPzl[10], hexPzl[11])  
H3 = (hexPzl[12], hexPzl[13], hexPzl[14], hexPzl[15], hexPzl[16], hexPzl[17], hexPzl[18])  
H4 = (hexPzl[19], hexPzl[20], hexPzl[21], hexPzl[22], hexPzl[23])  
P1 = (hexPzl[12], hexPzl[5], hexPzl[6], hexPzl[0], hexPzl[1]) *# P1 = positive diagonal*P2 = (hexPzl[13], hexPzl[14], hexPzl[7], hexPzl[8], hexPzl[2], hexPzl[3])  
P3 = (hexPzl[20], hexPzl[21], hexPzl[15], hexPzl[16], hexPzl[9], hexPzl[10], hexPzl[4])  
P4 = (hexPzl[22], hexPzl[23], hexPzl[17], hexPzl[18], hexPzl[11])  
N1 = (hexPzl[5], hexPzl[12], hexPzl[13], hexPzl[19], hexPzl[20]) *# N1 = negative diagonal*N2 = (hexPzl[0], hexPzl[6], hexPzl[7], hexPzl[14], hexPzl[15], hexPzl[21], hexPzl[22])  
N3 = (hexPzl[1], hexPzl[2], hexPzl[8], hexPzl[9], hexPzl[16], hexPzl[17], hexPzl[23])  
N4 = (hexPzl[3], hexPzl[10], hexPzl[11], hexPzl[18])

checkCopies = {hexTL, hexTR, hexML, hexMC, hexMR, hexBL, hexBR,  
 P1, P2, P3, P4, N1, N2, N3, N4}   
 *# checkCopies = all the different constraints to check* **for** tpl **in** checkCopies:  
 **if** copies(tpl):  
 **return True** *# true if the overall puzzle contains a subhex with copies, i.e. it's invalid*