COP 2930 Exam #1 Day #2 Solutions
2/21/2020

General Directions: For this portion of the exam you will write/complete 5 programs. Please create your own variables as necessary and follow the other directions given in each problem.

1) (10 pts) A six pack of chicken nuggets cost 2.99 and a nine pack of chicken nuggets cost 3.99. Complete the program below so that it asks the user how many six packs and how many nine packs they want to buy and prints out their total cost. Assume no sales tax. Please use the variables created that store 2.99 and 3.99.

```
COST6 = 2.99
COST9 = 3.99

num6 = int(input("How many 6 packs do you want to buy?\n"))
num9 = int(input("How many 9 packs do you want to buy?\n"))

cost = num6*COST6 + num9*COST9

print("Your total cost with tax is", cost)
```

Grading: 2 pts total for both int(input's.
6 pts for the cost calculation
2 pts for printing the result

2) (12 pts) You must buy either all 6 packs of nuggets or all 9 packs of nuggets. The cost of the packs are the same as in the previous problem. Depending on how many nuggets you need, it may be cheaper to buy all 6 packs or all 9 packs. (For example, if you are buying 12 nuggets, it's better to buy 2 6 packs than 2 nine packs, but if you are buying 100 nuggets, it's better to buy 12 9 packs than 17 6 packs. Note in this latter case, if you buy the 9 packs, you'll end up with 108 nuggets, which is okay, since you are getting at least 100.) Complete the program below so that it calculates the minimum cost for buying *n* (or more) chicken nuggets in this manner, where *n* is entered by the user. The method of solution will be to first figure out how many 6 packs are necessary and how many 9 packs are necessary, followed by calculating the total cost of both methods, followed by comparing which cost is less and outputting a message to that effect.

```
COST6 = 2.99
COST9 = 3.99

n = int(input("How many chicken nuggets do you want to buy?\n"))

num6packs = n//6

if n%6 > 0 :
    num6packs += 1

num9packs = n//9

if n%9 > 0 :
    num9packs += 1

cost6packs = num6packs*COST6

cost9packs = num9packs*COST9

if cost6packs < cost9packs :

     print("It is cheaper to buy", num6packs, "6 packs.")
     print("The total cost will be", cost6packs)

else:

     print("It is cheaper to buy", num9packs, "9 packs.")

     print("The total cost will be", cost9packs)
```

Grading: 1 pt int(input
2 pts if n%6, 2 pts if n%9
2 pts cost 6, 2 pts cost 9
1 pt if comparison
1 pt printing cost6
1 pt printing cost9

3) (12 pts) Now consider being allowed to buy any number of 6 packs and any number of 9 packs to obtain at least n chicken nuggets, where n is entered by the user. To figure out the best strategy, a chart would be useful which prints out the following information:

(a) # of 6 packs, (b) # of 9 packs, (c) total # of nuggets, and (d) total cost

We can do this buy looping through the total number of 6 packs (so the first row of the chart prints a combination of no 6 packs and all 9 packs, the next row prints a combination of 1 6 pack and the rest 9 packs and so forth. We can reuse the code from question 2 that calculates how many 6 or 9 packs we need to buy at least n nuggets for this solution. Complete the program below to solve this task:

```
COST6 = 2.99
COST9 = 3.99

n = int(input("How many chicken nuggets do you want to buy?\n"))

max6packs = n//6

if n%6 > 0 :
    max6packs += 1

for num6packs in range(max6packs+1):

    nuggetsNeeded = n - 6*num6packs

    if nuggetsNeeded < 0:

         nuggetsNeeded = 0

    num9packs = nuggetsNeeded//9

    if nuggetsNeeded%9 > 0 :
        num9packs += 1

    totalNuggets = 6*num6packs + 9*num9packs

    totalCost = num6packs*COST6 + num9packs*COST9

    print(num6packs,num9packs,totalNuggets,totalCost)
```

Grading: **1 pt int(input,** **2 pts n%6 > 0**
2 pts max6packs+1, **2 pts n - 6*num6packs**
1 pt nuggetsNeeded%9 > 0 **2 pts 6*n6+9*n9**
2 pts n6*C6 + n9*C9

4) (15 pts) Use the Python Turtle **AND A LOOP** to draw 25 horizontal lines, each of length 400 pixels with a horizontal spacing of 20 pixels with the bottom most line connecting the (x,y) locations (-200, -250) and (200, -250) and the top most line connecting the (x,y) locations (-200, 250) and (200, 250). To help you see the pattern, here is a list of the first five line segments and last five line segments that should be drawn:

First Five	Last Five
(-200, -250) to (200, -250)	(-200, 170) to (200, 170)
(-200, -230) to (200, -230)	(-200, 190) to (200, 190)
(-200, -210) to (200, -210)	(-200, 210) to (200, 210)
(-200, -190) to (200, -190)	(-200, 230) to (200, 230)
(-200, -170) to (200, -170)	(-200, 250) to (200, 250)

(Hint: Run a for loop to go through all the unique y coordinates, starting at -250, skipping 20, and ending at 250, inclusive. To draw each line, pick up the turtle pen, move to the correct starting position for that line segment on the left side of it, put the pen down, and then move forward the appropriate number of pixels. Your program should be 5 lines long.)

Relevant Turtle Commands

```
turtle.penup() - Pulls up the pen. No drawing when moving.
turtle.pendown() - Pulls down the pen. Drawing when moving.
turtle.forward(distance) - Moves turtle forward distance pixels
                           in the direction the turtle is headed.
turtle.setpos(x,y) - Moves the turtle to (x, y).
```

Program Goes Here

```
import turtle

for y in range(-250, 275, 25):
    turtle.penup()
    turtle.setpos(-200, y)
    turtle.pendown()
    turtle.forward(400)
```

Grading: 3 per line

11) (1 pt) What fruit is found in a blueberry bagel? **Blueberry** (Grading: give to all)