SI@UCF Introduction to Programming in Python with Recursion Test #2 Solutions
6/28/2017

1) (14 pts) A BOGO (Buy One Get One) is a deal at grocery stores where if you buy an item at regular price, you get a second one of that item for free. For example, if Gatorade was on a BOGO and one 32 oz. bottle of Gatorade cost $1.99, if you wanted to buy 5 of them, you would just pay $5.97. (If you buy 2, you get 2 free and then you just have to pay full price for the third.) Complete the program below so that it asks the user to enter the number of items they want, their regular price, and the sales tax as a percentage, and prints out the final purchase price. (Note: First calculate the cost without tax and then apply the tax on this total and just print out the final result. Also the sales tax percentage may not be an integer. Orange County's sales tax is 6.5%)

```
def main():

    numitems = int(input("How many items?\n"))
    cost = float(input("What is the cost of a one item?\n"))
    taxperc = float(input("What's the sales tax percentage?\n"))

    total = numitems//2*cost

    if numitems%2 == 1:
        total += cost
    total = total*(1 + taxperc/100)
    print("Your grand total is $", total)

main()
```

Grading: 1 pt each for int, float, float, 2 pts //2, 1 pt *cost, 4 pts for odd case, 3 pts for tax, 1 pt print

2) (15 pts) Write a segment of code which prints a pyramid of numbers that count up and down on each row. Specifically, on row i, with i starting at 1, the numbers listed will go from 1 to i, and then back down to 1. For example, a pyramid of 4 rows would look like this:

```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

```
n = int(input("How many rows do you want?\n"))

for i in range(n-1,-1, -1):

    for j in range(2*i):
        print(" ", end="")

    top = n - i
    for j in range(1,top+1):
        print(j, end=" ")
    for j in range(top-1,0, -1):
        print(j, end=" ")

    print()
```

Grading: 3 pts outer loop, 4 pts spaces, 4 pts counting up, 4 pts counting down

3) (15 pts) Write a recursive function that takes in the following parameters: `mylist, low, high, target` and returns the number of values in the list `mylist`, in between indexes `low` and `high`, inclusive, that are less than or equal to target. Please assume that `mylist` stores integers sorted from lowest to highest.

```
def numAtOrBelow(mylist, low, high, target):

    if low > high:
        return 0

    mid = (low+high)//2
    if target < mylist[mid]:
        return numAtOrBelow(mylist, low, mid-1, target)
    else:
        return mid-low+1+numAtOrBelow(mylist,mid+1,high,target)
```

Grading: 3 pts base case, 3 pts calc mid, 3 pts rec call for left, 6 pts for right side - 3 pts for adding mid-low+1, 3 pts for rec call

4) (15 pts) Write a function below that takes in a list called nums that stores integers and returns the range of the list. The range of a list is defined as the maximum value in the list minus the minimum value in the list. For example, the range of [3, 9, 2, 8, 12, 6, 5, 6] is 12 - 2 = 10.

```
def getRange(nums):

    mymin = nums[0]
    mymax = nums[0]

    for i in range(1,len(nums)):
        if nums[i] > mymax:
            mymax = nums[i]
        if nums[i] < mymin:
            mymin = nums[i]

    return mymax - mymin
```

Grading: 1 pt set to first value, 3 pts for loop, 4 pts for max update, 4 pts min update, 2 pts subtract, 1 pt return

5) (15 pts) Write a recursive function that prints out all odometer settings where the digits of the odometer are in *strictly increasing* order. Your function should take in the following parameters: `mylist, k, numD`, where `mylist` stores the odometer with the first `k` digits fixed and the number of different digits in use is `numD`. For example, if `mylist` stores `[2,3,-1,-1]` and `k=2` and `numD=8`, the recursive function should print `[2,3,4,5], [2,3,4,6], [2,3,4,7], [2,3,5,6], [2,3,5,7], [2,3,6,7]`

```
def printIncOdom(mylist, k, numD):

    if k == len(mylist):
        print(mylist)
        return

    start = 0
    if k > 0:
        start = mylist[k-1]+1

    for i in range(start, numD):
        mylist[k] = i
        printIncOdom(mylist, k+1, numD)
```

Grading: 4 pts case case print(2 if, 1 print, 1 return or else), 4 pts setting start for loop, 2 pts loop, 2 pts set mylist[k], 3 pts rec call

6) (15 pts) Consider writing a program that keeps track of items in a grocery store. One way to implement storing how many of each item the store has in stock would be to use a dictionary, where the key is the item and the value is the number of that item in stock. For example, the dictionary `{'soup': 5, 'pasta':12, 'candy':100, 'milk':17}` has 5 soup in stock, 12 pasta in stock, 100 candy in stock and 17 milk in stock. Assume that a dictionary called **`storestock`** stores this information. Write a segment of code that asks the user to enter the item they want to buy, and how many of that item they want to buy. Your segment of code should respond by printing `"In Stock"` if the store has the user's request in stock or `"Out of Stock"`, if the store does not have enough quantity of the item in question. Keep in mind that the user may request an item that isn't in the store at all (ie. not a valid key in the dictionary) and your segment of code must not crash if the user requests an item not in the store at all. Your program should simply print `"Out of Stock"` in this case.

```
item = input("What do you want to buy?\n")
quantity = int(input("How many of "+item+" do you want?\n"))

if item in storestock and quantity <= storestock[item]:
    print("In Stock")
else:
    print("Out of Stock")
```

Grading: 5 pts input, 3 pts check in dictionary, 3 pts check quantity, 4 pts rest

7) (10 pts) Explain in words what the following "game" in pyGame does:

```
import pygame, sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((600, 600))
pygame.display.set_caption("Test 2 Question")
black   = pygame.Color(0  ,0  ,0  )
purple  = pygame.Color(255,0  ,255)
red     = pygame.Color(255,0  ,0  )
x1  = 300
y1  = 300
dx1 = 5
dy1 = 5
x2 = 100
y2 = 50
dx2 = 0
dy2 = 0
pts = 0

while True:

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

        if event.type == KEYDOWN:
            if event.key == K_DOWN:
                y1 += dy1
            if event.key == K_UP:
                y1 -= dy1
            if event.key == K_LEFT:
                x1 -= dx1
            if event.key == K_RIGHT:
                x1 += dx1

    mydx = x1 - x2
    mydy = y1 - y2
    mag = (mydx**2 + mydy**2)**.5

    if mag < 2:
        break

    movex = int(mydx/mag*5)
    movey = int(mydy/mag*5)
    x2 += movex
    y2 += movey

    DISPLAYSURF.fill(black)
    pygame.draw.circle(DISPLAYSURF, purple, (x1, y1), 20, 0)
    pygame.draw.circle(DISPLAYSURF, red, (x2, y2), 20, 0)
    pygame.display.update()
    pygame.time.wait(60)
    pts += 1

print("You score",pts,"points.")
```

Answer for #7 here

This game has the user controlling the purple ball with the arrow keys. There isn't a "velocity" stored, so the ball only moves once when the user presses the key. There is a second ball, a red ball, that is chasing the purple ball. (Specifically, it does this by looking at the direction it needs to go to catch the purple ball at a moment in time, and heading in that direction, roughly 5 pixels total.) The game ends with the red ball is within 2 pixels of the purple ball and the user's score is simply the number of game frames they've been able to elude being caught by the red ball.

Grading: Mention of two balls - 2 pts, Mention that the user controls the purple one - 2 pts, Mention that the red one is chasing the purple one - 3 pts, Mention when the game ends - 1 pt, Mention what the score represents - 2 pts

8) (1 pt) What senator and congressman are responsible for revising the Frank-Dodd Wall Street Reform and Consumer Protection Act?

Barney Frank (MA) and Chris Dodd (CT), Grading: give to all

Python Random

```
random.seed() - seeds the random number generator
random.randint(a,b) - returns a random integer in between a and
                      b, inclusive.
```

Python List Methods

```
list.append(x) - adds x to the end of list
len(x) - returns the length of list x

How to create an empty list:
items = []
```

PyGame Drawing

```
# Draws a circle on the display surface SURF with center (x,y)
# radius r with color c and thickness t. If t=0, it's filled in.
pygame.draw.circle(SURF, c, (x,y), r, t)
```

Key constants

```
K_DOWN - key for down arrow
K_UP - key for up arrow
K_LEFT - key for left arrow
K_RIGHT - key for right arrow
```