Spring 2020 COP 2930 Final Exam Part B - Solutions

Directions: Write a single python program to solve each problem posed. No main function necessary. For questions that ask to write a function, write the function and test it with the main function provided. The name for your file will be provided within the parentheses at the beginning of the question. Please upload each of your .py files into the appropriate turn in, in Webcourses.

1) (10 pts) **(money.py)** Write a program that reads in the number of cents the user has and the prints out how much money they have in dollars and cents, such that the number of cents listed is in between 0 and 99, inclusive. (For example, if the user entered 347, your program should print out 3 dollars and 47 cents.) In order to earn full credit, your program can NOT have a loop.

Sample Solution
```
cents = int(input("How many cents do you have?\n"))
print("That is",cents//100,"dollars and",cents%100,"cents.")
```

Grading: 5 pts for each item.

2) (10 pts) **(stayhome.py)** With the novel coronavirus circulating, it's strongly recommended that individuals 65 or older OR with pre-existing conditions (asthma, diabetes, heart disease, obesity) always stay at home, since these individuals are at higher risk for complications. Write a program that asks the user to enter their age and also asks if they have any pre-existing conditions (no need to list these in your question), and then prints out, "Please stay at home always," if the user meets either or both conditions, or prints out, "You may leave the house for limited important activities."

Sample Solution
```
age = int(input("How old are you?\n"))
conditions = input("Do you have any pre-existing
conditions(yes/no)?\n")

if age >= 65 or conditions == "yes":
    print("Please stay at home always")
else:
    print("You may leave the house for limited important
activities.")
```

Grading: 2 pts for each read, 4 pts for if, 1 pt for each print

3) (20 pts) **(foursquare.py)** Anya and Arup have been playing lots of four square at home recently! Help them visualize their playing field by writing a program that draws a sample four square field of arbitrary size. Your program should ask the user to enter a single positive integer, *n*, the size of a single square. Your design should be 2*n*+3 rows and 2*n*+3 columns. If you label your rows and columns from 0 to 2*n*+2, then all characters on rows 0, *n*+1 and 2*n*+2 should be stars and all characters on columns 0, *n*+1 and 2*n*+1 should also be stars. All other characters should be spaces. Here is the design for *n* = 2:

```
*******
*  *  *
*  *  *
*******
*  *  *
*  *  *
*******
```

In order to earn credit your program must (a) have a nested loop structure, (b) NOT use python's ability to "multiply" a string or character, and (c) must work for any positive integer value of n less than 40 but be less than 40 lines long!

Sample Solution

```
n = int(input("Please enter the size of each square.\n"))

for i in range(2*n+3):
    for j in range(2*n+3):

        if i%(n+1) == 0 or j%(n+1) == 0:
            print("*", end="")
        else:
            print(" ", end="")

    print()
```

Grading: 4 pts right # of rows, 4 pts right # of columns, 4 pts stars in only necessary rows, 4 pts stars in only necessary columns, 4 pts advance properly to next line

4) (10 pts) (**harmonic.py**) Write a function that takes in a single positive integer *n*, and returns the nth Harmonic number. The nth Harmonic number is the sum of all the reciprocals of the first n integers. For example, the third Harmonic number is $1 + \frac{1}{2} + \frac{1}{3} \sim 1.83$. (Note: You can check the last test case…the 100th Harmonic number is roughly 5.187.)

```
def harmonic(n):
    # Code goes here
def main():
    print(harmonic(1))
    print(harmonic(3))
    print(harmonic(100))
main()
```

Sample Solution

```
def harmonic(n):

    total = 0

    for i in range(1,n+1):
        total += 1.0/i

    return total
```

Grading: 1 pt set accumulator to 0, 4 pts loop, 4 pts add fraction, 1 pt return

5) (25 pts) (**donations.py**) A food bank receives donations, one by one. Write a program that reads in a list of each item a food bank receives from the file **donations.txt**. After reading in each item, your program should print out two things on a line: the item received, as well as the quantity of that item received thus far. The first line of the input file will have a single positive integer, n, the number of donations received. Each of the following n lines will contain the next donation received. All donations will be uppercase strings. Here is a sample text file:

donations.txt (Sample File)

```
10
BEANS
APPLES
ORANGES
BREAD
BREAD
MEAT
APPLES
APPLES
ORANGES
PASTA
```

In order to receive credit, you must use a dictionary that maps items to the number of that item in stock in your program. Here is sample output for the corresponding input file:

```
BEANS 1
APPLES 1
ORANGES 1
BREAD 1
BREAD 2
MEAT 1
APPLES 2
APPLES 3
ORANGES 2
PASTA 1
```

Sample Solution

```
file = open("donations.txt", "r")
numItems = int(file.readline().strip())
itemDictionary = {}

for i in range(numItems):

    item = file.readline().strip()

    if item in itemDictionary:
        itemDictionary[item] = itemDictionary[item] + 1
    else:
        itemDictionary[item] = 1

    print(item,itemDictionary[item])

file.close()
```

Grading: 4 pts opening file, 2 pts reading num items, 2 pts setting up dictionary, 2 pts looping through items, 2 pts reading one item, 2 pt check in dictionary, 4 pts add 1 in this case, 4 pts setting to 1 in other case, 3 pts print