**10/15/15**

'''

Donato,Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

assignment6Ex1

'''

'''

Analysis:

Generate income tax and net income given the gross income and marital

status of the user.

Output to monitor:

maritalStatus (str) - The marital status of the user.

income (int) - The taxable income of the user (in dollars).

taxes (float) - The amount of tax the person owes (in dollars).

total (float) - The net income of the user (in dollars).

Input from keyboard:

maritalStatus (str) - The marital status of the user.

income (int) - The taxable income of the user (in dollars).

Tasks allocated to Functions:

computeTax()

netIncome()

incomeValidation()

maritalStatusValidation()

'''

#Constants

INCOME\_9075 = 9075

INCOME\_18150 = 18150

INCOME\_36900 = 36900

INCOME\_73800 = 73800

INCOME\_89350 = 89350

INCOME\_148850 = 148850

INCOME\_186350 = 186350

INCOME\_226850 = 226850

INCOME\_405100 = 405100

INCOME\_406750 = 406750

INCOME\_457600 = 457600

TAXED\_INCOME\_907\_5 = 907.5

**Page 1 of 10**

TAXED\_INCOME\_1815 = 1815

TAXED\_INCOME\_5081\_25 = 5081.25

TAXED\_INCOME\_10162\_5 = 10162.5

TAXED\_INCOME\_18193\_75 = 18193.75

TAXED\_INCOME\_28925 = 28925

TAXED\_INCOME\_45353\_75 = 45353.75

TAXED\_INCOME\_50765 = 50765

TAXED\_INCOME\_109587\_5 = 109587.5

TAXED\_INCOME\_117541\_25 = 117541.25

TAXED\_INCOME\_118118\_75 = 118118.75

TAXED\_INCOME\_127962\_5 = 127962.5

TAX\_PERCENT\_10 = .10

TAX\_PERCENT\_15 = .15

TAX\_PERCENT\_25 = .25

TAX\_PERCENT\_28 = .28

TAX\_PERCENT\_33 = .33

TAX\_PERCENT\_35 = .35

TAX\_PERCENT\_396 = .396

# Computes income tax

# param maritalStatus (str)

# param income (int)

# returns tax owed (float)

def computeTax(maritalStatus, income):

if (maritalStatus == "single" or maritalStatus == "Single"):

if (income >= 0 and income <= INCOME\_9075):

tax = (TAX\_PERCENT\_10 \* income)

elif income <= INCOME\_36900:

tax = (TAXED\_INCOME\_907\_5 + (TAX\_PERCENT\_15 \* income))

elif income <= INCOME\_89350:

tax = (TAXED\_INCOME\_5081\_25 + (TAX\_PERCENT\_25 \* income))

elif income <= INCOME\_186350:

tax = (TAXED\_INCOME\_18193\_75 + (TAX\_PERCENT\_28 \* income))

elif income <= INCOME\_405100:

tax = (TAXED\_INCOME\_45353\_75 + (TAX\_PERCENT\_33 \* income))

elif income <= INCOME\_406750:

tax = (TAXED\_INCOME\_117541\_25 + (TAX\_PERCENT\_35 \* income))

else:

tax = (TAXED\_INCOME\_118118\_75 + (TAX\_PERCENT\_396 \* income))

**Page 2 of 10**

else:

if (income >= 0 and income <= INCOME\_18150):

tax = (TAX\_PERCENT\_10 \* income)

elif income <= INCOME\_73800:

tax = (TAXED\_INCOME\_1815 + (TAX\_PERCENT\_15 \* income))

elif income <= INCOME\_148850:

tax = (TAXED\_INCOME\_10162\_5 + (TAX\_PERCENT\_25 \* income))

elif income <= INCOME\_226850:

tax = (TAXED\_INCOME\_28925 + (TAX\_PERCENT\_28 \* income))

elif income <= INCOME\_405100:

tax = (TAXED\_INCOME\_50765 + (TAX\_PERCENT\_33 \* income))

elif income <= INCOME\_457600:

tax = (TAXED\_INCOME\_109587\_5 + (TAX\_PERCENT\_35 \* income))

else:

tax = (TAXED\_INCOME\_127962\_5 + (TAX\_PERCENT\_396 \* income))

return tax

# Computes net income

# param income (int)

# param tax (float)

# returns the net income (float)

def netIncome(income, tax):

total = (income - tax)

return total

# Validates income

# param income (str)

# return True when income is only made of digits and greater than 0

def incomeValidation(income):

return income.isdigit() and int(income) > 0

# Validates marital status

# param maritalStatus (str)

# return True when marital status is single or married

def maritalStatusValidation(maritalStatus):

return maritalStatus == 'single' or maritalStatus == 'Single'\

or maritalStatus == 'married' or maritalStatus == 'Married'

# The program calculates the income tax and the net income of the user

# given marital status and gross income

def main():

# Priming read

name = input("Please enter your name or press <Enter> to quit:")

**Page 3 of 10**

while name:

# Takes in the users marital status

maritalStatus = input("Are you single or married? ")

# Validates the marital status

while not maritalStatusValidation(maritalStatus):

print("You did not enter a valid marital status. \

Please enter either single or married.")

maritalStatus = input("Are you single or married? ")

# Takes in the users income

income = input("What is your taxable income? (in dollars) ")

# Validates the income

while not incomeValidation(income):

print("You did not enter a valid income. Please enter your income.")

income = (input("What is your taxable income? (in dollars) "))

# Coverts income into an int

income = int(income)

# Computes income tax

tax = computeTax(maritalStatus, income)

# Computes net income

total = netIncome(income, tax)

# Displays marital status, gross income, and taxes owed

print("You've entered that you're",maritalStatus,"and that you make $", \

income,"a year. Therefore your income tax is: $", tax)

# Displays net income

print("Your net income would be:$", total)

# Continuation of the loop

name = input("Please enter your name or press <Enter> to quit:")

'''

# tester

# Test values should show 'smooth' transition from one tax bracket to the \

next

**Page 4 of 10**

states = ['single', 'married']

incomes = [[0,9075, 9076, 36900, 36901, 89350, 89351,

186350, 186351, 405100, 405101, 406750, 406751],

[0, 18150, 18151, 73800, 73801, 148850, 148851,

226850, 226851, 405100, 405101, 457600, 457601]]

for i in range(len(states)):

for j in range(len(incomes[0])):

print("%s, $%.2f = $%.2f" % \

(states[i], incomes[i][j], computeTax(states[i], incomes[i][j])))

'''

main()

'''

Donato,Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

assignment6Ex2

'''

'''

Analysis:

Create a game where the player inputs the amount of money he or she is

willing to bet and that is added to the pot. The program then simulates

a dice roll for each turn and if the sum of the dice is 7, the player

wins $4. If not, the player loses $1. The aformentioned amounts are

added or subtracted from the pot as the program runs. The game ends when

there is no money left in the pot.

Output to monitor:

rollNumber (int) - What number roll it is.

diceValue (int) - Sum of the dice.

pot (int) - The amount of money in the pot.

bestPot (int) - The most amount of money in the pot.

bestRoll (int) - The number roll that the user had the most money in

the pot.

**Page 5 of 10**

Input from keyboard:

pot (int) - The starting value of the pot.

Tasks allocated to Functions:

diceSum()

luckySevensGame()

initalPotValidation()

'''

import random

# Constants

RANGE\_OF\_DICE\_1 = 1

RANGE\_OF\_DICE\_7 = 7

MONEY\_ADDED\_TO\_POT = 4

DICE\_ROLL\_OF\_7 = 7

# Validates starting pot value

# param initalPot (str)

# return True when the starting pot is only made of digits and greater

# than 0

def initalPotValidation(initalPot):

return initalPot.isdigit() and int(initalPot) > 0

# Simulates rolling 2 dice

# returns the sum of two die

def diceSum():

dice1 = random.randrange(RANGE\_OF\_DICE\_1,RANGE\_OF\_DICE\_7)

dice2 = random.randrange(RANGE\_OF\_DICE\_1,RANGE\_OF\_DICE\_7)

diceSum = (dice1 + dice2)

return diceSum

**Page 6 of 10**

# Plays the Lucky Sevens Game

# param pot (int)

# returns the roll, the value of the roll, the amount of money in the pot

# the best roll and the most money in the pot

def luckySevensGame(pot):

rollNumber = 0

bestRoll = 0

bestPot = pot

while pot > 0:

diceValue = diceSum()

if diceValue == DICE\_ROLL\_OF\_7:

pot = pot + MONEY\_ADDED\_TO\_POT

else:

pot = pot - 1

rollNumber += 1

if pot > bestPot:

bestPot = pot

bestRoll = rollNumber

print(rollNumber, '\t', diceValue, '\t', pot)

print("Your best roll was roll", bestRoll, "and you had", bestPot, \

"dollars.")

print("You lost all your money after", rollNumber, "rolls.")

# The program takes in the inital betting value and then plays the Lucky

# Sevens Game.

def main():

# Priming read

pot = input("Please enter a starting bet in whole dollars\

or press <Enter> to quit. ")

while pot:

# Validates the users inital pot input

while not initalPotValidation(pot):

print("Your input is incorrect.\

Please enter an amount in whole dollars.")

pot = input("Please enter a starting bet in whole dollars\

or press <Enter> to quit.")

# Prints the head of the table

print("Roll",'\t',"Value", '\t', "Dollars in Pot")

# Converts the inital pot value into the

pot = int(pot)

**Page 7 of 10**

# Plays the Lucky Sevens Game

rounds = luckySevensGame(pot)

# Continuation of the loop

pot = input("Please enter a starting bet in whole dollars\

or press <Enter> to quit. ")

main()

'''

Donato,Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

assignment6Ex3

'''

'''

Analysis:

Create a program that counts the amount of money a person would earn over

a period of time if his or her salary is one penny the first day and

doubles each day.

Output to monitor:

dayCount (int) - The number day it is in the computation

daySalary (int) - The amount of money earned that day

finalDayCount (int) - The last day the person earned money

salaryTotal (int) - The total amount of money made

Input from keyboard:

days (int) - The amount of days the program runs for

Tasks allocated to functions:

daysValidation()

salaryComputation()

'''

# Constants

DOUBLES = 2

ONE\_CENT = .01

**Page 8 of 10**

# Validates the day value

# param days (str)

# return True when the str is composed of only numbers and is greater

# than 1

def daysValidation(days):

return days.isdigit() and int(days) > 0

# Computes the salary

# param days (int)

# returns each day the computation occurs for, the daily salary on a

# given day and the total number of days the computation goes on for

# and the final total pay

def salaryComputation(days):

dayCount = 1

daySalary = ONE\_CENT

salaryTotal = 0

while dayCount <= days:

print(dayCount,'\t', '\t','\t', '\t', '\t','\t', daySalary)

salaryTotal = salaryTotal + daySalary

daySalary = (daySalary \* DOUBLES)

dayCount = dayCount + 1

finalDayCount = dayCount - 1

print("In", finalDayCount,"days, a penny grows to $",salaryTotal)

# The program takes in the number of days a person would calculate for

# and then determines the amount of money the person would make if his

# or her salary was one cent the first day and then doubled each day

def main():

# Priming input

days = input("Enter the number of days for which the salary is to be \

computed. Or press <Enter> to quit: ")

while days:

#Validates the users inital day input

while not daysValidation(days):

print("Invalid number of days. Please enter a valid number of days \

to calculate.")

days = input("Enter the number of days for which the salary is to be \

computed. Or press <Enter> to quit: ")

# Converts the day value into int

days = int(days)

**Page 9 of 10**

# Starts setting up the table

print("Day", '\t', '\t','\t', '\t', '\t','\t', "Pay")

print("-----------------------------------------------------")

# Determines the total amount of money made and the money made each

# day

salaryComputation(days)

# Continuation of the loop

days = input("Enter the number of days for which the salary is to be \

computed. Or press <Enter> to quit: ")

main()

**Page 10 of 10**