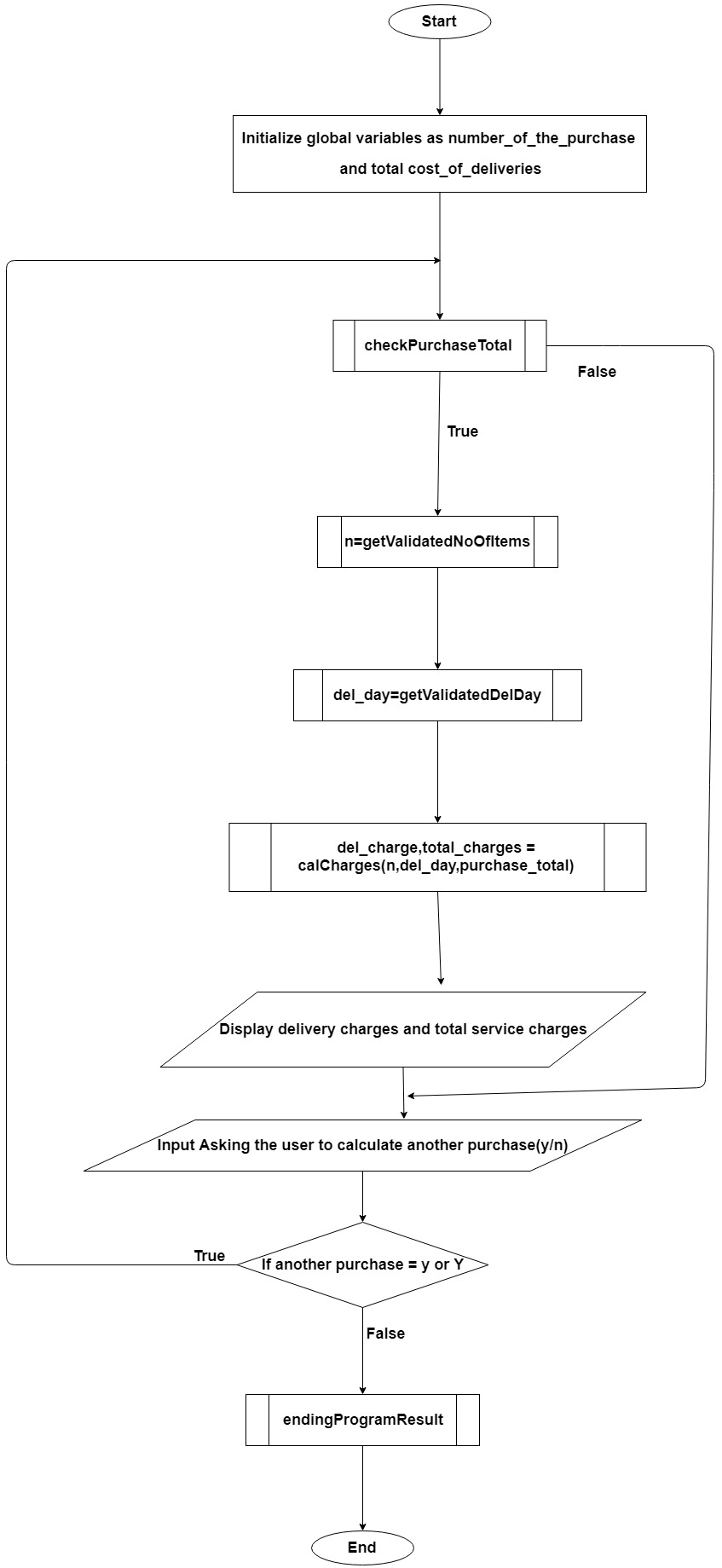
Assignment 2

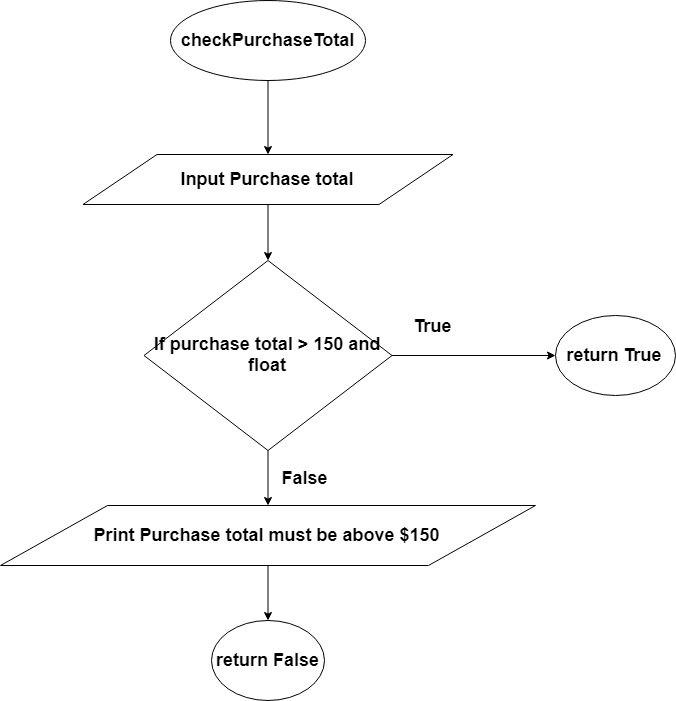
ITC558 - Programming Principles

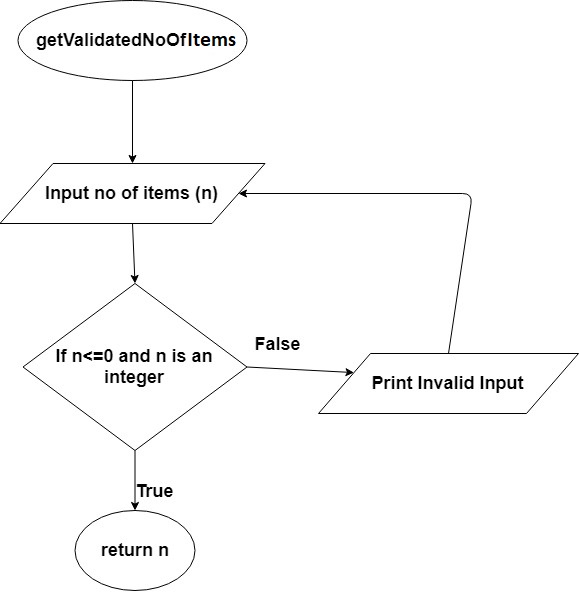
Submitted By:

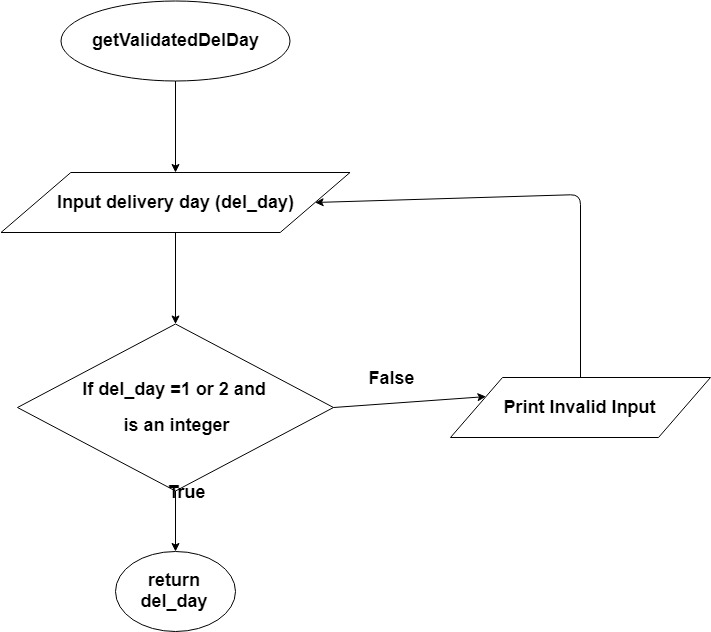
Sudeep Dahal

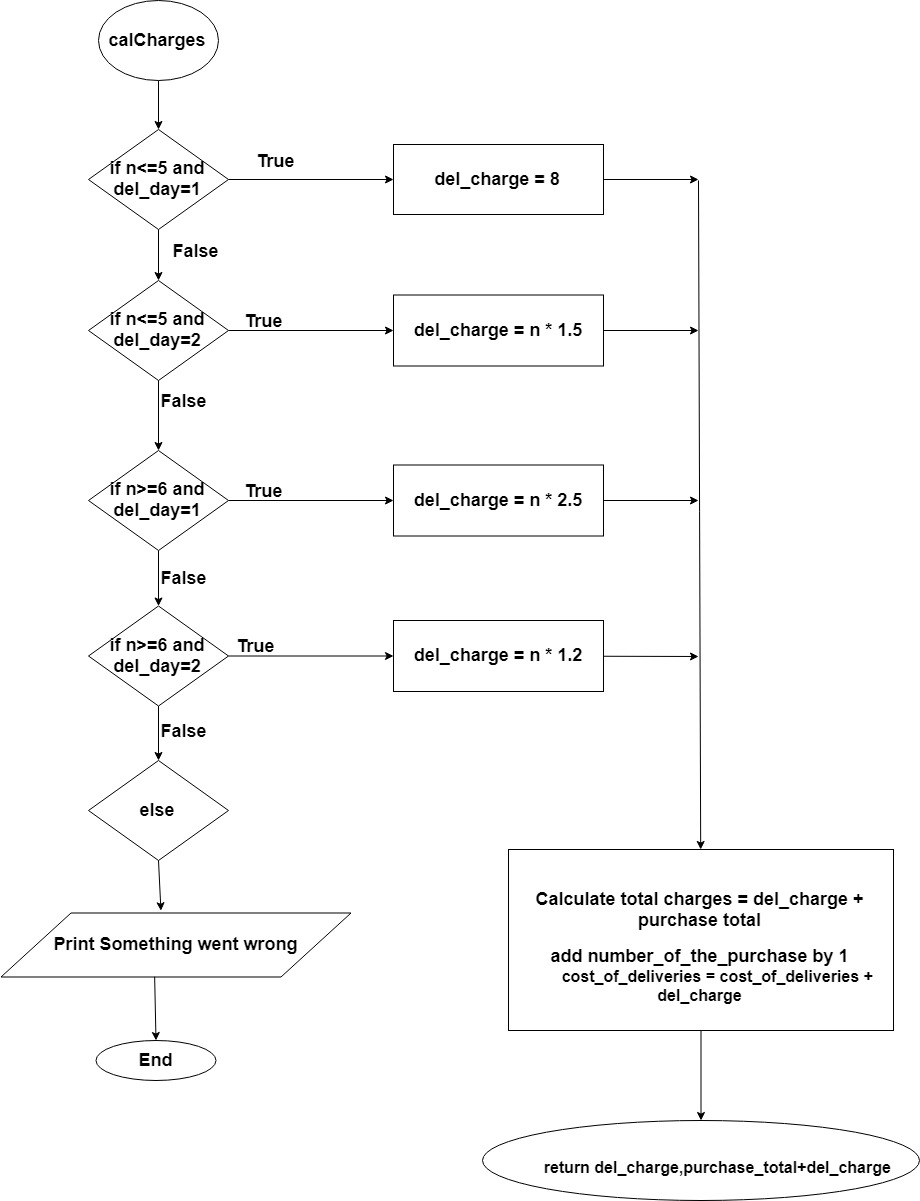
11668527

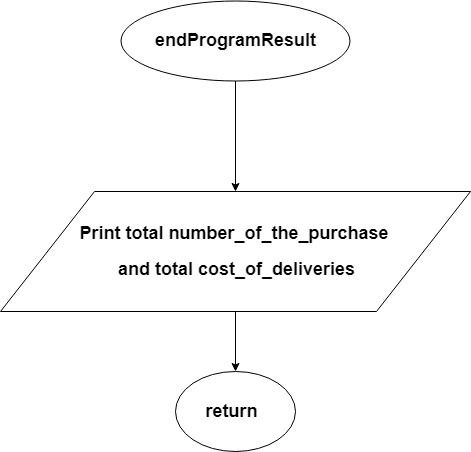












Above Figures shows the flowchart of the developed program

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Data Table** | | | | |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** | **The screenshot of actual output when the test data is used** |
| Normal | Purchase total = 198,  No. of items = 6,  Delivery day =,1 | Normal valid integer values (To check normal process) | 15.00, 213.00 | Figure 1 |
| Normal | Purchase total = 305,  No. of items = -6 / 4,  Delivery day = -1/ 1 | Testing for validation of negative numbers | 8.00, 313.00 | Figure 2 |
| Normal | Purchase total = s / 198,  No. of items = s / 6,  Delivery day = s / 1 | Testing for string input | Display invalid input message and request to input again | Figure 3 |
| Abnormal | Purchase total = ‘ ’ / 555,  No. of items = ‘ ’ / 8,  Delivery day = ‘ ’ / 2 | Testing empty value input | Display invalid input message and request to input again | Figure 4 |
| Abnormal | Purchase total = $ / 198,  No. of items = # / 6,  Delivery day = ! / 1 | Testing special character input | Display invalid input message and request to input again | Figure 5 |

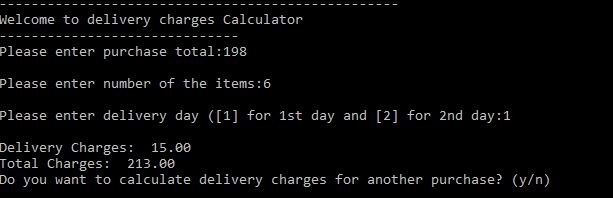


Figure 1

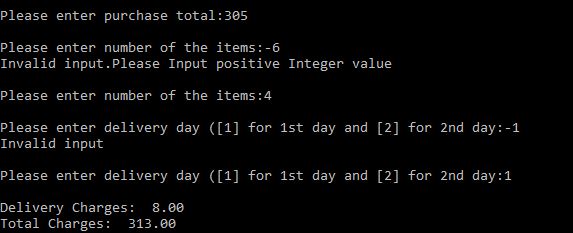


Figure 2

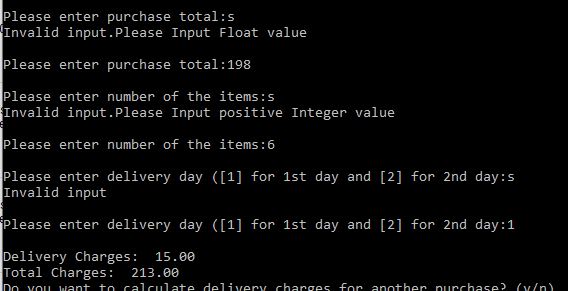


Figure 3

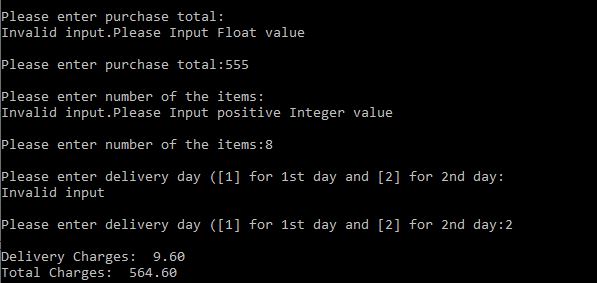


Figure 4

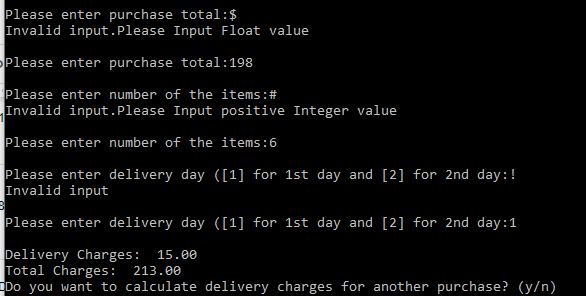


Figure 5

**Source Code:**

#counter to count the total number of purchase and delivery total

number\_of\_the\_purchase = 0

cost\_of\_deliveries = 0

def main():

#code for upper header design

designCode()

#To checking for users next purchase

another\_purchase = 'y'

#Initializing delivery charge, purchase total and number of items to 0

del\_charge = 0.0

purchase\_total = 0.0

n = 0

#Repeating the delivery calculator if user enters y or Y

while (another\_purchase == 'y' or another\_purchase == 'Y'):

purchase\_total = None

#Validating to get float value

while type(purchase\_total) is not float:

try:

purchase\_total = float(input('\nPlease enter purchase total:'))

except ValueError:

print("Invalid input.Please Input Float value")

#checking for purchase total is above 150 or not and returns boolean value

purchaseTotal = checkPurchaseTotal(purchase\_total)

if purchaseTotal:

#call function to get validate input and store in n variable

n = getValidatedNoOfItems()

#call function to get validate input and store in del\_day variable

del\_day = getValidatedDelDay()

#Calling the calculating function

del\_charge,total\_charges = calCharges(n,del\_day,purchase\_total)

#Display of delivery charges and total charges as floating-point numbers with 2 decimal places

print('\nDelivery Charges: ',format(del\_charge,'.2f'))

print('Total Charges: ',format(total\_charges,'.2f'))

else:

#else case if purchase total is below $150

print('ERR: Sorry, purchase total need to be above $150.')

#Asking if users wants to use the delivery calculator again

another\_purchase = input('Do you want to calculate delivery charges for another purchase? (y/n)')

#Printing number of delivery and total cost of delivery

endingProgramResult()

def designCode():

for x in range(50):

print('-',end='')

print('\nWelcome to delivery charges Calculator')

for x in range(30,):

print('-',end='')

def checkPurchaseTotal(purchase\_total):

if purchase\_total > 150:

return True

else:

return False

def getValidatedNoOfItems():

#check if input is integer or not

n = None

#Validating to get positive int value

while True:

try:

n = int(input('\nPlease enter number of the items:'))

except ValueError:

print("Invalid input.Please Input positive Integer value")

continue

else:

if (n<=0):

print("Invalid input.Please Input positive Integer value")

continue

else:

return n

break

def getValidatedDelDay():

#check if input is integer or not and is 1 or 2

del\_day = None

while True:

try:

del\_day = int(input('\nPlease enter delivery day ([1] for 1st day and [2] for 2nd day:'))

except ValueError:

print("Invalid input")

continue

else:

if (del\_day == 1 or del\_day == 2):

return del\_day

else:

print("Invalid input")

continue

def calCharges(n,del\_day,purchase\_total):

#calculation of delivery charges accordingly

if n <= 5 and del\_day == 1:

del\_charge = 8.0

elif n <= 5 and del\_day == 2:

del\_charge = n \* 1.5

elif n >= 6 and del\_day == 1:

del\_charge = n \* 2.5

elif n >= 6 and del\_day == 2:

del\_charge = n \* 1.2

else:

#If non of the case behave accordingly exiting the program

print('Something went wrong')

exit()

global number\_of\_the\_purchase,cost\_of\_deliveries

number\_of\_the\_purchase += 1

cost\_of\_deliveries += del\_charge

return del\_charge,purchase\_total+del\_charge

def endingProgramResult():

print('\nThanks for using the delivery charges Calculator!\nSee you again!\n')

print('Total number of purchase made: ',number\_of\_the\_purchase)

print('Total cost of Delivery made: ',format(cost\_of\_deliveries,'.2f'))

main()

Source Code is also at [https://github.com/sudeepdahal/Py\_a1/blob/master/Assignment2.py](https://github.com/sudeepdahal/Py_a1/blob/master/Assignment1.py)

Source Code File:



References:

Draw.io. (2019). *Flowchart Maker & Online Diagram Software*. [online] Available at: https://www.draw.io/ [Accessed 16 Aug. 2019].

W3schools.com. (2019). *Python Tutorial*. [online] Available at: https://www.w3schools.com/python/ [Accessed 15 Aug. 2019].

Codecademy. (2019). *Python Tutorial: Learn Python For Free | Codecademy*. [online] Available at: https://www.codecademy.com/learn/learn-python [Accessed 16 Aug. 2019].

(2019). Retrieved 15 September 2019, from <https://www.quora.com/How-can-I-make-sure-the-user-inputs-a-positive-integer-in-Python>

Python Tip: Validating user input as number (Integer) | 101 Computing. (2019). Retrieved 15 September 2019, from <https://www.101computing.net/number-only/>