Problem Set - Functions Pass By Value

1. In a while loop - allow the user to enter a quantity and price. Use a function to compute the total (quantity times price). The function should be passed the quantity and price and then return the total. In the function, provide a 10% discount if the total is over $10,000.00. Display quantity, price and total. Sum and display the extended price.

Input	Process	Output

	CompExtPrice(qty, unitprice)
	    Extprice = qty*unitprice
	
	    If extprice > 10000
	        Discamt = extprice * 0.10
	    Else
	        Discamt = 0
	
	    newExtPrice = extPrice – discamt
	
	    return newExtPrice	

Qty		Extprice

price	Main
	    totalExtPrice = 0
	
	    Do you want to do this program (Yes or No)
	
	    While (Yes)
	        Input qty, price
	        Extprice = CompExtPrice(qty,price)
	        Display qty, price, Extprice
	        totalExtPrice = totalExtPrice + extprice
	        Do you want to continue with this program?	

	Display totalExtPrice	totalExtPrice

2. Using a for loop for a team of 9 players - enter players last name, number of hits and at bats at the keyboard. Use a function to compute batting average. Pass the hits and at bats to the function. The function should return batting average. Display last name and batting average. Give a count of the number of players entered.

Input	1. **Player Last Name**: Entered by the user (string). 2. **Number of Hits**: Total hits made by the player (integer). 3. **At Bats**: Total at-bats (integer).

Process	1. **Loop through players**: Use a `for` loop to collect data for a team of 9 players. 2. **Function for batting average**: Pass `hits` and `at-bats` to the function, which calculates and returns `batting average = hits / at-bats` (if at-bats > 0).
	3. **Store and count players**: Save each player's last name and batting average in a list. Maintain a count of players processed.
Output	1. **Player batting averages**: Display each player's last name and batting average. 2. **Player count**: Display the total number of players (9).

3. Use a loop – (your choice) Enter the destination city, miles traveled, and gallons used for a trip. Use a function to compute miles per gallon. Pass miles travelled and gallons used to the function. The function should return miles per gallon. Count the number of entries made (number of trips) Display destination city, miles and mpg. At end display the number of entries made.

Input	1. **Destination city**: The name of the city entered by the user. 2. **Miles traveled**: Distance traveled in miles (numeric input). 3. **Gallons used**: Fuel used (numeric input)
Process	1. **Initialize variables**: Start with a trip count of 0 and an empty list to store trip details. 2. **Loop**: Ask the user for input until they choose to stop.
	- Get destination city, miles traveled, and gallons used from the user.
	Function call: Pass miles and gallons to a function (compute_mpg) that calculates miles per gallon (mpg = miles / gallons).
	- Store the trip details (city, miles, MPG) in a list.
	- Increment the trip count.
Output	1. **Trip summary**: For each trip, display: - Destination city - Miles traveled - Miles per gallon (MPG). 2. **Total trips**: Display the total number of trips entered.

4. Use a loop -Allow the employee to enter last name, job code and hours worked. Use a function to determine the pay rate. Pass to this function the job code and it should return rate of pay. Use Job code L is $25/hr, A is $30/hr and J is $50/hr for respective pay rates. Compute gross pay. Give time and a half for overtime. Display last name and gross pay. Sum and display total of all gross pay.

Input	1. **Last Name**: The employee's last name. 2. **Job Code**: Job classification (`L`, `A`, or `J`). 3. **Hours Worked**: Total hours worked by the employee (numeric input).
Process	1. **Loop to accept inputs**: Repeat until the user decides to stop. 2. **Determine pay rate**: Use a function (`get_pay_rate`) to return the hourly rate based on the job code.

	3. **Calculate gross pay**: - If hours worked > 40, compute overtime as `(hours worked - 40) * 1.5 * pay rate`. - Add regular pay `(40 * pay rate)` if applicable.
	- For 40 hours or fewer, compute as `hours worked * pay rate`. 4. **Store employee data**: Keep track of each employee's last name and gross pay.
	5. **Summarize total gross pay**: Accumulate gross pay for all employees into a running total.
Output	1. **Employee details**: For each employee, display: - Last name - Gross pay 2. **Summary statistics**: Display the total gross pay for all employees.

5. Use a loop - Allow the user to enter student last name, credit hours and district code. Use a function to compute tuition owed. Charge In district (code of I) $250 per credit hour. Out of district (code of O) is $550 per credit hour. The function should receive credit hours and district code and return tuition owed. Display student name and tuition owed. Sum and display total of all tuition owed.

Input	1. **Student Last Name**: Entered by the user (string). 2. **Credit Hours**: Number of credit hours the student is taking (numeric input). 3. **District Code**: Code indicating in-district (`I`) or out-of-district (`O`) (string).
Process	1. **Loop to accept inputs**: Repeatedly prompt the user for student details until they decide to stop.
	2. **Compute tuition owed**: Use the `compute_tuition` function, passing credit hours and district code. The function returns tuition based on: - `I`: $250 per credit hour. - `O`: $550 per credit hour. - Invalid codes return `None`.
	3. **Store details**: Save each student's last name and tuition owed in a list. 4. **Accumulate total tuition**: Add each student's tuition to a running total.
Output	1. **Individual tuition details**: Display each student's name and tuition owed. 2. **Summary**: Display the total tuition owed for all students.

Examples - Optional

1. Enter the number of Points and redemption code. For redemption code C then compute value as 2 x rewards points. Redemption code X then they get 3 x rewards points. All other codes get 1.5 x rewards points. Write a function that receives points and redemption code and computes rewards points. Display points, redemption code and rewards points.
2. Enter two numbers and operation code (A, S, M, D). Write a function that receives the two numbers and uses the operation code to perform an operation on the two numbers

(A=addition, S=Subtraction, M=Multiplication, D=Division). Check for dividing by 0. If the second number is 0 then set result to -999. Display two number, operation code, result and message if attempt to divide by zero.

3. Allow the user to enter a string. The string can be entered with any case (all upper, all lower of mixed). Write a function that accepts the string and returns all lower case when the original string is all upper or mixed. If the original string is all lower then make the string all upper case. The function should return the new string. Display both the original and new string.