Problem Set - Functions Pass By Value

1. Allow the user to enter a quantity and price, use ctl+z to stop. 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 $10,0000.00. 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 |
|  |  |  |
|  |  |  |

1. Enter players last name, number of hits and at bats at the keyboard, use ctl+z to stop. 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 | Process | Output |
| Player's last name | Continuously prompt the user to enter the player’s last name, number of hits, and number of at-bats. | Player's last name and calculated batting average displayed. |
| Number of hits and at-bats | For each player, pass the number of hits and at-bats to a function designed to calculate the batting average. | Total number of players entered. |
| User stops the input process using **Ctrl+Z** | The function computes the batting average using the formula: batting average = hits / at-bats. |  |
|  | Display the player’s last name along with the calculated batting average. |  |
|  | Maintain a count of the number of players entered throughout the process. |  |

1. Enter the destination city, miles travelled and gallons used for a trip, use ctl+z to stop. 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 | Process | Output |
| Destination city name | Continuously prompt the user to enter the destination city, miles traveled, and gallons used for each trip. | Destination city, miles traveled, and miles per gallon (MPG) displayed for each trip |
| Miles traveled | For each entry, pass the miles traveled and gallons used to a function that calculates miles per gallon. | Total count of trips entered. |
| Gallons used | computes miles per gallon using the formula: miles per gallon = miles traveled / gallons used.d |  |
| User stops the input process using **Ctrl+Z** | Display the destination city, miles traveled, and the calculated miles per gallon (MPG) |  |
|  | Maintain a count of the number of trips entered during the process. |  |

1. Allow the employee to enter last name, job code and hours worked, use ctl+z to stop. 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 | Process | Output |
| Last Name, Job Code, Hours Worked | total\_gross\_pay to 0 | Employee Last Name, Gross Pay |
| Job Code (L, A, or J) | enter last\_name, job\_code, and hours\_worked until Ctrl+Z is pressed. | Running total of Gross Pay |
| Hours Worked | Jobcode  Get pay rate  'L' → $25/hr  'A' → $30/hr  'J' → $50/hr  Calculate gross pay:regular\_pay = 40 \* pay\_rate overtime\_pay = (hours\_worked - 40) \* (pay\_rate \* 1.5) If hours worked ≤ 40, gross pay = hours\_worked \* pay\_rateIf hours worked > 40, gross pay = regular\_pay + overtime\_pay | Final Total Gross Pay |
|  | last\_name  gross\_pay |  |
|  | Accumulate each gross\_pay into total\_gross\_pay. |  |
|  | After the loop ends, display total\_gross\_pay. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Allow the user to enter student last name, credit hours and district code, use ctl+z to stop. 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 | Process | Output |
| Last Name, Credit Hours, District Code | compute\_tuition\_owed(credit\_hours, district\_code) to compute tuition | Display student last name and tuition owed. |
| District Code ('I' or 'O') | If district code is 'I': tuition = credit\_hours \* $250. If district code is 'O': tuition = credit\_hours \* $550. | Running total of all tuition owed. |
| Credit Hours | Add the computed tuition owed to the total. | Display total tuition owed. |
| **Do you want to continue? (Yes/No)** | **If Yes, repeat the process for another student. If No, stop.** |  |
|  |  |  |

Examples

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.