**Problem 1: Online Shopping Cart**

An e-commerce platform allows users to add items to their shopping cart. The program should calculate the total price based on quantity and price per item. However, errors may occur if:

* The quantity or price is not a valid number.
* The user enters a negative value for quantity or price.

Write a Python program that:

1. Accepts item name, quantity, and price per item from the user.
2. Calculates the total amount.
3. Uses try-except to handle invalid input (non-numeric or negative values).
4. Raises a custom exception (InvalidValueError) if negative values are entered.
5. Displays appropriate error messages and ensures the program does not crash.

**Program:**

try:

item = input("Enter item name: ")

quantity = int(input("Enter quantity: "))

price = float(input("Enter price per item: "))

if quantity < 0 or price < 0:

raise ValueError("Quantity and price must be non-negative.")

total = quantity \* price

print("Total amount for", item, "is", total, "rupees")

except ValueError:

print("Error Type: ValueError - Invalid numeric input or negative value.")

except Exception:

print("Error Type: General Exception - Unexpected error occurred.")

else:

print("Order processed successfully!")

finally:

print("Thank you for shopping with us!")

**Problem 2: Bank Account Withdrawal**

A bank application allows customers to withdraw money from their account. The system must ensure that:

* The withdrawal amount is not greater than the available balance.
* The input is numeric and positive.

Write a Python program that:

1. Has an initial account balance (e.g., ₹10,000).
2. Accepts the withdrawal amount from the user.
3. Uses try-except-else-finally blocks to:
   * Handle ValueError if the input is not numeric.
   * Raise a custom exception (InsufficientBalanceError) if the withdrawal amount exceeds the balance.
   * Print “Transaction Successful” if no exceptions occur.
   * Display “Thank you for banking with us” in the finally block.

**Program:**

balance = 10000 # initial balance

try:

print("Your current balance is:", balance)

amount = float(input("Enter amount to withdraw: "))

if amount <= 0:

raise ValueError("Amount must be positive.")

if amount > balance:

raise ValueError("Insufficient balance to complete the transaction.")

balance = balance - amount

print("Withdrawal successful! Remaining balance:", balance)

except ValueError:

print("Error Type: ValueError - Invalid or incorrect withdrawal amount.")

except Exception:

print("Error Type: General Exception - Unexpected error occurred.")

else:

print("Transaction completed successfully!")

finally:

print("Thank you for banking with us!")

**Problem 3: Student Grade Management System**

A program reads student marks from a dictionary and calculates their grades. However, there may be issues such as:

* Missing subjects or marks.
* Marks outside the valid range (0–100).

Write a Python program that:

1. Stores student marks in a dictionary like {"Math": 90, "Science": 105, "English": 80}.
2. Uses try-except to catch KeyError when a subject is missing.
3. Raises a custom exception (InvalidMarksError) if marks are outside 0–100.
4. Displays the average and grade if all values are valid.

**Program:**

try:

student\_marks = {"Math": 90, "Science": 85, "English": 80}

total = 0

subjects = ["Math", "Science", "English"]

for subject in subjects:

marks = student\_marks[subject] # Access marks of each subject

if marks < 0 or marks > 100:

raise ValueError("Marks for " + subject + " must be between 0 and 100.")

total = total + marks

average = total / len(subjects)

if average >= 90:

grade = 'A'

elif average >= 75:

grade = 'B'

elif average >= 60:

grade = 'C'

else:

grade = 'D'

print("Average Marks:", average)

print("Grade:", grade)

except KeyError:

print("Error Type: KeyError - Missing subject data.")

except ValueError:

print("Error Type: ValueError - Invalid marks entered.")

except Exception:

print("Error Type: General Exception - Unexpected error occurred.")

else:

print("Grade calculation completed successfully!")

finally:

print("End of student record processing.")