Exception Handling – Answer

# Section A:

1. Write a program to divide two numbers entered by the user. Handle ZeroDivisionError using try-except.

try:  
 a = int(input("Enter numerator: "))  
 b = int(input("Enter denominator: "))  
 result = a / b  
 print("Result:", result)  
except ZeroDivisionError:  
 print("Error: Division by zero is not allowed.")

2. Write a program to convert a string to an integer. Handle ValueError if the input is not a valid number.

try:  
 user\_input = input("Enter a number: ")  
 number = int(user\_input)  
 print("Converted integer:", number)  
except ValueError:  
 print("Error: Invalid input, not a number.")

3. Accept two numbers from the user and perform addition. Use try-except to handle invalid input types.

try:  
 a = float(input("Enter first number: "))  
 b = float(input("Enter second number: "))  
 print("Sum:", a + b)  
except ValueError:  
 print("Error: Please enter valid numbers.")

4. Write a program to read an element from a list using an index entered by the user. Handle IndexError.

my\_list = [10, 20, 30, 40, 50]  
try:  
 index = int(input("Enter index: "))  
 print("Element at index:", my\_list[index])  
except IndexError:  
 print("Error: Index out of range.")

# Section B:

1. Create a program that accepts a number from the user and prints its square. Use try-except-else to handle ValueError.

try:  
 num = int(input("Enter a number: "))  
except ValueError:  
 print("Invalid input. Please enter a valid number.")  
else:  
 print("Square:", num \* num)

2. Write a program to open a file and read contents. Use try-except-else to handle FileNotFoundError.

try:  
 with open("sample.txt", "r") as file:  
 content = file.read()  
except FileNotFoundError:  
 print("Error: File not found.")  
else:  
 print("File content:\n", content)

3. Write a Python program to convert a number to its binary format. Use try-except-else to handle any invalid input.

try:  
 num = int(input("Enter an integer: "))  
except ValueError:  
 print("Error: Not a valid integer.")  
else:  
 print("Binary format:", bin(num))

# Section C:

1. Write a program that opens a file and ensures it gets closed, whether or not an exception occurs.

try:  
 file = open("example.txt", "r")  
 content = file.read()  
 print(content)  
finally:  
 file.close()  
 print("File closed.")

2. Simulate a login process where the user input is handled in a try block and a log message is printed in finally.

try:  
 username = input("Enter username: ")  
 password = input("Enter password: ")  
 if username == "admin" and password == "1234":  
 print("Login successful.")  
 else:  
 print("Login failed.")  
finally:  
 print("Login attempt logged.")

3. Write a program that divides two numbers, catching errors with try-except, and printing a clean-up message using finally.

try:  
 a = int(input("Enter numerator: "))  
 b = int(input("Enter denominator: "))  
 print("Result:", a / b)  
except ZeroDivisionError:  
 print("Error: Cannot divide by zero.")  
finally:  
 print("Division operation attempted.")

# Section D:

1. Create a program that handles multiple exceptions: ZeroDivisionError, ValueError, and always prints "Execution complete" using finally.

try:  
 a = int(input("Enter numerator: "))  
 b = int(input("Enter denominator: "))  
 print("Result:", a / b)  
except ZeroDivisionError:  
 print("Error: Cannot divide by zero.")  
except ValueError:  
 print("Error: Invalid input.")  
finally:  
 print("Execution complete.")

2. Write a program to simulate bank withdrawal. Use try-except-else-finally to handle incorrect amount input.

balance = 5000  
try:  
 amount = int(input("Enter withdrawal amount: "))  
 if amount > balance:  
 raise ValueError("Insufficient balance.")  
except ValueError as e:  
 print("Transaction failed:", e)  
else:  
 balance -= amount  
 print("Transaction successful. Remaining balance:", balance)  
finally:  
 print("Thank you for using our service.")