Lab Assignment 7.2

Program	**: B. Tech (CSE)**
Course Title	**: AI Assisted Coding**
Course Code	**: 23CS002PC304**
Semester	**III**
Academic Session	**: 2025-2026**
Name of Student	**: K. Ruthwik Bhat**
Enrollment No.	**: 2403A51L08**
Batch No.	**51**
Date	**: 30-01-2026**

Task-1: Runtime Error Due to Invalid Input Type

Buggy Code:

```
num = input("Enter a number: ")
result = num + 10
print(result)
```

Prompt: Identify the cause of the runtime error and modify the program

Corrected Code:

```
num = input("Enter a number: ")
result = num + 10
print(result)
```

Output:

```
Enter a number: 10
20
```

Explanation:

- The runtime error occurs because the input function returns a string, and we are trying to add an integer (10) to a string (num).
- By converting the input to an integer using int(), we ensure that both operands in the addition are of the same type (integers), allowing the addition to be performed without errors.

Task-2: Incorrect Function Return Value

Buggy Code:

```
def square(n):
    result = n * n
```

Prompt: A function is designed to calculate the square of a number, but it does not return the computed result properly. So, correct the code to return the result.

Corrected Code:

```
def square(n):
    result = n * n
    return result
print(square(5))
```

Output:

```
25
```

Explanation:

- The function does not have a return statement, so it does not return the computed result.
- By adding the return statement, the function will now return the computed square of the number when called.

Task-3: IndexError in List Traversal

Buggy code:

```
numbers = [10, 20, 30]
for i in range(0, len(numbers)+1):
print(numbers[i])
```

Prompt: A Python program iterates over a list using incorrect index limits, causing an IndexError. Identify the error and correct the code.

Corrected Code:

```
numbers = [10, 20, 30]
for i in range(0, len(numbers)):
    print(numbers[i])
```

Output:

```
10
20
30
```

Explanation:

- The error occurs because the range function is set to go up to len(numbers)+1, which causes the loop to attempt to access an index that is out of bounds for the list.
- By changing the range to len(numbers), we ensure that the loop only iterates over valid indices of the list, preventing the IndexError from occurring.

Task-4: Uninitialized Variable Usage

Buggy Code:

```
if True:
    pass
print(total)
```

Prompt: A program uses a variable in a calculation before assigning it any value. Identify the error and modify the code to fix it.

Corrected Code:

```
total = 0
if True:
    total += 10
print(total)
```

Output:

```
10
```

Explanation:

- The error occurs because the variable 'total' is used before it has been assigned a value.
- By initializing 'total' to 0 before using it, we ensure that it has a defined value when we perform the calculation, preventing any runtime errors.

Task-5: Logical Error in Student Grading System

Buggy Code:

```
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "C"
else:
    grade = "B"
print(grade)
```

Prompt: A grading program assigns incorrect grades due to improper conditional logic. Identify the logical error and correct the code.

Corrected code:

```
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "B"
else:
    grade = "C"
print(grade)
```

Output:

```
B
```

Explanation:

- The logical error in the code is that the conditions for assigning grades are not in the correct order. The condition for grade “C” is checked before the condition for grade “B” which causes students with marks between 80 and 89 to be incorrectly assigned a “C” instead of a “B”.
- By reordering the conditions, we ensure that students with marks between 80 and 89 are correctly assigned a “B”, while those with marks below 80 receive a “C”.