SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week4 - Wednesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:8.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases **Lab Objectives:** • To introduce students to test-driven development (TDD) using AI code generation tools. • To enable the generation of test cases before writing code implementations.	Week4 - Wednesday

- To reinforce the importance of testing, validation, and error handling.
- To encourage writing clean and reliable code based on AI-generated test expectations.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Use AI tools to write test cases for Python functions and classes.
- Implement functions based on test cases in a test-first development style.
- Use unittest or pytest to validate code correctness.
- Analyze the completeness and coverage of AI-generated tests.
- Compare AI-generated and manually written test cases for quality and logic

Task Description#1

Use AI to generate test cases for is_valid_email(email) and then implement the validator function.

Requirements:

- Must contain @ and . characters.
- Must not start or end with special characters.
- Should not allow multiple @.

Expected Output#1

- Email validation logic passing all test cases

Task Description#2 (Loops)

- Ask AI to generate test cases for assign_grade(score) function. Handle boundary and invalid inputs.
  Requirements
- AI should generate test cases for assign_grade(score) where: 90-100: A, 80-89: B, 70-79: C, 60-69: D, <60: F
- Include boundary values and invalid inputs (e.g., -5, 105, "eighty").

Expected Output#2

Grade assignment function passing test suite

Task Description#3

- Generate test cases using AI for is_sentence_palindrome(sentence). Ignore case, punctuation, and spaces

  Requirement
- Ask AI to create test cases for is_sentence_palindrome(sentence)
  (ignores case, spaces, and punctuation).

- Example:
  "A man a plan a canal Panama" → True

Expected Output#3

- Function returns True/False for cleaned sentences
- Implement the function to pass AI-generated tests.

Task Description#4

- Let AI fix it Prompt AI to generate test cases for a ShoppingCart class (add_item, remove_item, total_cost).

  Methods:
  Add_item(name,orice)
  Remove_item(name)

Total_cost()

Expected Output#4

- Full class with tested functionalities

Task Description#5

- Use AI to write test cases for convert_date_format(date_str) to switch from "YYYY-MM-DD" to "DD-MM-YYYY".
  Example: "2023-10-15" → "15-10-2023"

Expected Output#5

- Function converts input format correctly for all test cases

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Task #1	0.5
Task #2	0.5
Task #3	0.5
Task #4	0.5
Task #5	0.5
Total	**2.5 Marks**

TASK-1:

FUNCTION WITH MY TESTCASES:

task1.py > ...

```python
import re

def is_valid_email(email):
    """
    Verifies if the given email address is valid.
    Returns True if valid, False otherwise.
    """
    pattern = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(pattern, email) is not None
#test case-1
print(is_valid_email("namitha@gmail.com"))  # Expected output: True
#test case-2
print(is_valid_email("namitha@gmailcom"))  # Expected output: False
#test case-3
print(is_valid_email("namitha@.com"))  # Expected output: False
```

OUTPUT:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/task1.py
True
False
False
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8>
```

AI GENERATED TESTCASES:

task1.py | test_task-1.py | task3.py | test_task-3.py | task4.py | test_task-4.py | task2.py

test_task-1.py > ...

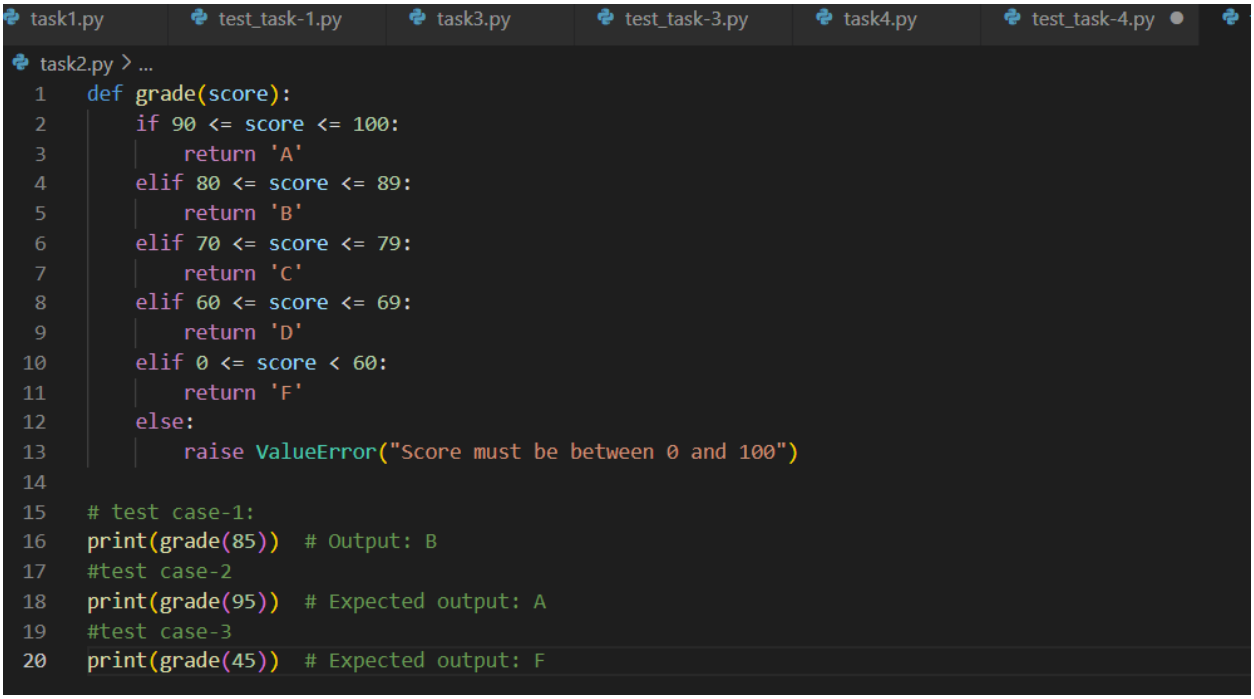
```python
import unittest
from task1 import is_valid_email

class TestIsValidEmail(unittest.TestCase):
    def test_valid_email(self):
        self.assertTrue(is_valid_email("test@example.com"))
        self.assertTrue(is_valid_email("user.name@domain.co"))
        self.assertTrue(is_valid_email("user_name123@sub.domain.org"))

    def test_invalid_email_no_at(self):
        self.assertFalse(is_valid_email("testexample.com"))
        self.assertFalse(is_valid_email("user.name.domain.com"))

    def test_invalid_email_no_domain(self):
        self.assertFalse(is_valid_email("test@"))
        self.assertFalse(is_valid_email("user@.com"))

    def test_invalid_email_special_chars(self):
        self.assertFalse(is_valid_email("user!@domain.com"))
        self.assertFalse(is_valid_email("user#name@domain.com"))

    def test_invalid_email_multiple_at(self):
        self.assertFalse(is_valid_email("user@@domain.com"))

    def test_invalid_email_empty_string(self):
        self.assertFalse(is_valid_email(""))

if __name__ == "__main__":
    unittest.main()
```

OUTPUT;

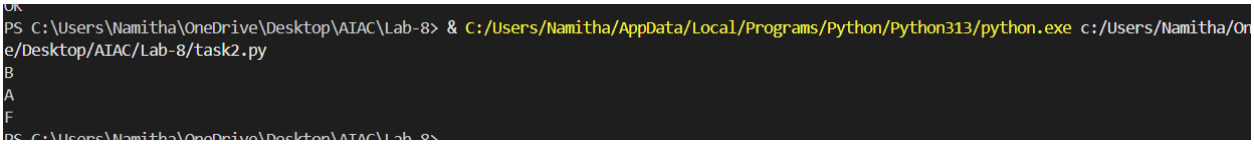
```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/test_task-1.py
True
False
False
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8>
```

TASK-2:

FUNCTION WITH MY TESTCASES:



OUTPUT:



AI GENERATED TESTCASES:

test_task-2.py > ...

```python
import unittest
from task2 import grade

class TestGradeFunction(unittest.TestCase):
    def test_grade_A(self):
        self.assertEqual(grade(95), 'A')
        self.assertEqual(grade(90), 'A')

    def test_grade_B(self):
        self.assertEqual(grade(85), 'B')
        self.assertEqual(grade(80), 'B')

    def test_grade_C(self):
        self.assertEqual(grade(75), 'C')
        self.assertEqual(grade(70), 'C')

    def test_grade_D(self):
        self.assertEqual(grade(65), 'D')
        self.assertEqual(grade(60), 'D')

    def test_grade_F(self):
        self.assertEqual(grade(59), 'F')
        self.assertEqual(grade(0), 'F')

    def test_invalid_input(self):
        with self.assertRaises(ValueError):
            grade(-1)
        with self.assertRaises(ValueError):
            grade(101)

if __name__ == '__main__':
    unittest.main()
```

OUTPUT:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDri
e/Desktop/AIAC/Lab-8/test_task-2.py
B
A
F
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8>
```

TASK-3:

FUNCTION WITH MY TESTCASES:

```python
import string

def is_sentence_palindrome(sentence):
    # Remove punctuation and spaces, convert to lowercase
    cleaned = ''.join(
        ch.lower() for ch in sentence if ch.isalnum()
    )
    return cleaned == cleaned[::-1]
#test case-1
print(is_sentence_palindrome("A man a plan a canal Panama"))  # Expected output: True
#test case-2
print(is_sentence_palindrome("Hello World"))  # Expected output: False
```

OUTPUT:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/task3.py
True
False
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8>
```

AI GENERATED TESTCASES:

```
t_task-3.py > TestIsSentencePalindrome > test_empty_string
import unittest
from task3 import is_sentence_palindrome

class TestIsSentencePalindrome(unittest.TestCase):
    def test_simple_palindrome(self):
        self.assertTrue(is_sentence_palindrome("A man a plan a canal Panama"))

    def test_with_punctuation(self):
        self.assertTrue(is_sentence_palindrome("Madam, in Eden, I'm Adam."))

    def test_not_palindrome(self):
        self.assertFalse(is_sentence_palindrome("Hello world"))

    def test_empty_string(self):
        self.assertTrue(is_sentence_palindrome(""))

    def test_single_character(self):
        self.assertTrue(is_sentence_palindrome("a"))

    def test_mixed_case(self):
        self.assertTrue(is_sentence_palindrome("No lemon, no melon"))

    def test_numbers_and_letters(self):
        self.assertTrue(is_sentence_palindrome("12321"))

    def test_spaces_only(self):
        self.assertTrue(is_sentence_palindrome("     "))

    def test_non_palindrome_with_punctuation(self):
        self.assertFalse(is_sentence_palindrome("This is not a palindrome!"))

if __name__ == "__main__":
    unittest.main()
```

OUTPUT:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/test_task-3.py
True
False
.........
----------------------------------------------------------------------
Ran 9 tests in 0.001s

OK
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8>
```

TASK-4:

FUNCTION WITH MY TESTCASES:

```python
class ShoppingCart:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price):
        self.items[name] = price

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]

    def total_cost(self):
        return sum(self.items.values())

#test case-1
cart = ShoppingCart()
cart.add_item("apple", 1.0)
cart.add_item("banana", 0.5)
print(cart.total_cost())  # Expected output: 1.5
cart.remove_item("apple")
print(cart.total_cost())  # Expected output: 0.5
#test case-2
cart2 = ShoppingCart()
cart2.add_item("milk", 2.0)
cart2.add_item("bread", 1.5)
print(cart2.total_cost())  # Expected output: 3.5
cart2.remove_item("bread")
print(cart2.total_cost())  # Expected output: 2.0
```

OUTPUT:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDrive/Desktop/AIAC/Lab-8/__pycache__/task4.py
1.5
0.5
3.5
2.0
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8>
```

AI GENERATED TESTCASES:

```
e__ > test_task-4.py > ...
import unittest

class ShoppingCart:
    def __init__(self):
        self.items = {}

    def add_item(self, name (parameter) price: Any
        self.items[name] = price

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]

    def total_cost(self):
        return sum(self.items.values())


class TestShoppingCart(unittest.TestCase):

    def test_add_and_total(self):
        cart = ShoppingCart()
        cart.add_item("apple", 1.0)
        cart.add_item("banana", 0.5)
        self.assertEqual(cart.total_cost(), 1.5)

    def test_remove_item(self):
        cart = ShoppingCart()
        cart.add_item("apple", 1.0)
        cart.add_item("banana", 0.5)
        cart.remove_item("apple")
        self.assertEqual(cart.total_cost(), 0.5)

    def test_update_item_price(self):
        cart = ShoppingCart()
        cart.add_item("chocolate", 2.0)
        cart.add_item("chocolate", 3.0)  # Overwrites old price
        self.assertEqual(cart.total_cost(), 3.0)
```

```python
    def test_remove_non_existent_item(self):
        cart = ShoppingCart()
        cart.add_item("juice", 1.5)
        cart.remove_item("candy")  # Not present
        self.assertEqual(cart.total_cost(), 1.5)

    def test_empty_cart_total(self):
        cart = ShoppingCart()
        self.assertEqual(cart.total_cost(), 0)

    def test_zero_price_item(self):
        cart = ShoppingCart()
        cart.add_item("free_sample", 0.0)
        self.assertEqual(cart.total_cost(), 0.0)

    def test_negative_price_item(self):
        cart = ShoppingCart()
        cart.add_item("discount_coupon", -5.0)
        self.assertEqual(cart.total_cost(), -5.0)

    def test_multiple_add_and_remove(self):
        cart = ShoppingCart()
        cart.add_item("pen", 1.0)
        cart.add_item("pencil", 0.5)
        cart.add_item("notebook", 2.5)
        self.assertEqual(cart.total_cost(), 4.0)
        cart.remove_item("pencil")
        self.assertEqual(cart.total_cost(), 3.5)
        cart.remove_item("pen")
        self.assertEqual(cart.total_cost(), 2.5)


if __name__ == "__main__":
    unittest.main()
```

Output:

```
e/Desktop/AIAC/Lab-8/__pycache__/task4.py
1.5
0.5
3.5
2.0
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/__pycache__/test_task-4.py
........
----------------------------------------------------------------------
Ran 8 tests in 0.002s
```

TASK-5:

FUNCTION WITH MY TEST CASES:

```
cache__ > task5.py > ...
    def convert_date_format(date_str):
        year, month, day = date_str.split("-")
        return f"{day}-{month}-{year}"

    # Example usages
    print(convert_date_format("2023-10-15"))  # Output: 15-10-2023
    print(convert_date_format("1999-01-05"))  # Output: 05-01-1999
```

Output:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/__pycache__/task5.py
15-10-2023
05-01-1999
```

AI GENERATED TESTCASES:

```
import unittest

def convert_date_format(date_str):
    year, month, day = date_str.split("-")
    return f"{day}-{month}-{year}"

class TestConvertDateFormat(unittest.TestCase):

    def test_valid_date(self):
        self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")

    def test_another_date(self):
        self.assertEqual(convert_date_format("1999-01-05"), "05-01-1999")

if __name__ == "__main__":
    unittest.main()
import unittest

def convert_date_format(date_str):
    year, month, day = date_str.split("-")
    return f"{day}-{month}-{year}"

import unittest

def convert_date_format(date_str):
    year, month, day = date_str.split("-")
    return f"{day}-{month}-{year}"

class TestConvertDateFormat(unittest.TestCase):

    def test_valid_date(self):...

    def test_another_date(self):
        self.assertEqual(convert_date_format("1999-01-05"), "05-01-1999")

if __name__ == "__main__":
    unittest.main()
```

```
    unittest.main()
class TestConvertDateFormat(unittest.TestCase):

    def test_valid_date(self):
        self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")

    def test_another_date(self):
        self.assertEqual(convert_date_format("1999-01-05"), "05-01-1999")

if __name__ == "__main__":
    unittest.main()
```

OUTPUT:

```
e/Desktop/AIAC/Lab-8/__pycache__/task5.py
15-10-2023
05-01-1999
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-8> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDriv
e/Desktop/AIAC/Lab-8/__pycache__/test_task-5.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```