Assignment 1 for Fundamentals and String Capabilities

Subject: Computer Science Workshop - 1 (CSE 2141)
Session: September 2025 to January 2026
Branch: Computer Science and Engineering (CSE)
Section: All
Course Outcomes: CO1
Program Outcomes: PO1, PO2, and PO5

Learning Levels: Remembering (L1), Understanding (L2), Application (L3).

Q no.	Questions	Learning Levels

Q1.	Define a Python function named `generate_bill(item, price, quantity=1, discount=0, tax_rate=0.05)` that calculates and prints the final payable amount for a purchased item after applying the discount and adding tax. The function should use default arguments for quantity, discount, and tax rate.
	The function should perform the following steps:
	• Compute the subtotal = price × quantity.
	• Apply a discount percentage on the subtotal.
	• Apply a tax (default = 5%) on the discounted amount.
	• Print a detailed bill summary showing item name, quantity, price, discount, tax, and total amount.
	Call the function in different ways:
	i. Using only the required arguments (`item` and `price`).
	ii. Providing a custom quantity while keeping default discount and tax.
	iii. Using named arguments for discount and tax while keeping default quantity.
	iv. Providing all arguments explicitly.
	Example:
	`Item: Laptop, Quantity: 2, Price: 50000, Discount: 10%, Tax: 5%`
	`Total Bill: 94500.0`
		L2, L3

Q2.	Design and implement a Python program that simulates a simple calculator using `if-elif` statements.
	The program should allow the user to choose an operation :— addition, subtraction, multiplication, division, or modulus and then input two numbers.
	The program should perform the selected arithmetic operation based on the user's choice and display the result in a clear and readable format.
	It should also handle division and modulus by zero using conditional checks and display an appropriate warning message.
	Input: Operation choice (add, sub, mul, div, mod) and two numbers entered by the user.
	Output: Display the result of the chosen operation or show a warning message if division or modulus by zero is attempted.
	Example: `Enter operation (add/sub/mul/div/mod): div`
	`Enter first number: 10`
	`Enter second number: 0`
	`Error: Division by zero not allowed.`
		L1, L2

Q no.	Questions	Learning Levels
Q3.	Design and implement a Python program that accepts the name of a month from the user as a string input. The program should determine and display the number of days in the specified month. For February, the program should identify the year (leap or non-leap year) first, before determining the number of days (28 or 29 days). Make use of `Dictionary` data structures for mapping months to their corresponding number of days. Handle both valid and invalid inputs by displaying an error message if the entered month name is incorrect. **Input:** Enter the name of a month: February Enter a year: 2024 **Output:** February 2024 has 29 days.	L2, L3
Q4.	Design and implement a Python program for two approaches for checking whether a given string is a palindrome using two separate functions. The **first function** should use a `for` loop to determine whether the given string is a palindrome. It should ignore only case differences (for example, "Madam" and "madam" should be treated as the same). The **second function** should use the **two-pointer technique** to check whether a string is a palindrome. This function should ignore case, spaces, and punctuation so that complete sentences such as "A man, a plan, a canal: Panama" can be correctly identified as palindromes. Finally, call both functions in the main program using the user input and display the results from both approaches for comparison. **Input:** A string entered by the user (e.g., "Madam") **Output:** Display the results from both palindrome-checking functions. For example: `For-loop Check: Palindrome, Two-pointer Check: Palindrome`	L2, L3
Q5.	Design and implement a Python program that reads a decimal number from the user and performs multiple base conversions. The program should: • Convert the decimal number into its **Binary**, **Octal**, and **Hexadecimal** representations using the built-in functions. • Display all three converted values without their prefixes (`0b`, `0o`, `0x`). • Count and display the number of digits in each converted representation. • Reverse the conversion — convert the binary, octal, and hexadecimal strings back to decimal using the `int(string, base)` function — and display the results to verify correctness. **Input:** A decimal number entered by the user (e.g., 255). **Output:** Display the binary, octal, and hexadecimal forms (without prefixes) and their digit counts. Then show the decimal values obtained by reconverting each representation.	L1, L2
Q6.	Design and implement a Python program that performs both encryption and decryption of a string. The encryption function should first reverse the input string and then swap every adjacent pair of characters to generate the encrypted text. The decryption function should reverse this process to obtain the original string. **Input:** A string entered by the user (e.g., "hello") **Output:** Display both the encrypted and decrypted strings. For example: `Encrypted: eholl, Decrypted: hello`	L2, L3

Q no.	Questions	Learning Levels
Q7.	Develop a Python program that reads a sentence from the user, splits it into words, sorts them in **reverse alphabetical order**, and joins them using a **custom separator** provided by the user. The program should ignore punctuation marks during processing so that only valid words are considered for sorting. **Input:** A sentence and a custom separator entered by the user (e.g., "Hello, world! Python.", "—"). **Output:** Display the sorted words joined by the specified separator. For example: `world–python–hello`	L2, L3
Q8.	Design and implement a Python function named `validate_password(password)` that checks whether a given password meets specific security requirements. The function should validate the password based on the following rules: • The password must contain at least **8 characters**. • It must include at least one **uppercase letter** (A–Z). • It must include at least one **lowercase letter** (a–z). • It must include at least one **digit** (0–9). • It must include at least one **special character** from the set `!@#$%` and no white-spaces. The function should return: • `True` if the password satisfies all the above conditions. • `False`, along with a list of specific error messages, if one or more rules are violated. **Input:** A string representing the password entered by the user (e.g., "Pass@123"). **Output:** Display whether the password is valid or invalid, and if invalid, list the violated rules.	L2, L3
Q9.	Develop a Python program that processes a paragraph of text entered by the user. The program should perform the following tasks: • Convert the entire paragraph into **title case** (each word starts with a capital letter). • Remove extra spaces between words using the `split()` and `join()` methods. • Count and display the occurrences of each **vowel** (A, E, I, O, U) using their character codes. **Input:** A paragraph entered by the user (e.g., " this is an example paragraph "). **Output:** Display the cleaned and title-cased paragraph, followed by the count of each vowel. For example: `Processed Text: This Is An Example Paragraph` `Vowel Counts → A: 3, E: 2, I: 1, O: 0, U: 0`	L2, L3
Q10.	Develop a Python program that counts the frequency of each word in a given sentence while ignoring case and punctuation. The program should process the sentence, remove punctuation marks, convert all words to lowercase, and then count how many times each unique word appears. Finally, display the word frequencies in a formatted string sorted alphabetically by word. **Input:** A sentence entered by the user (e.g., "Hello world, hello!"). **Output:** Display the sorted word frequencies in the format "word: count". For example: `hello: 2 world: 1`	L2, L3