Assignment-4 Regular Expressions
Subject: Computer Science Workshop - 1 (CSE 2141)
Session: September 2025 to January 2026
Branch: Computer Science and Engineering (CSE)
Section: All
Course Outcomes: CO1, CO2, CO3
Program Outcomes: PO1, PO2, PO3, and PO5
Learning Levels: Remembering (L1), Understanding (L2), Application (L3)

Q no.	Questions	Learning Levels
Q1.	Write a Python function filter_emails(lst) that takes a list of strings and returns **only** those strings that look like a simple email according to this simplified rule: • username: starts with a letter or digit, may contain letters, digits, dot (.) or underscore (_) afterwards • exactly one @ • domain: letters only (a–z or A–Z) followed by a dot and a 2–4 letter TLD (e.g. edu, com, in) **Input:** ["student123@gmail.com", "teacher.name@soa.edu", "abc.xyz@abc.in", "xyz#12@abc.com", "bad@1n.com"] **Output:** ["student123@gmail.com", "teacher.name@soa.edu", "abc.xyz@abc.in"]	L2, L3
Q2.	Write a Python function **verify_passwd(s)** that validates a password using regular expressions based on the following rules: 1. The password must be **at least 8 characters long**. 2. It may contain **letters, digits, or the special characters**: @ # $ % ^ & * ! 3. It must contain **at least one digit**. 4. It must contain **at least one letter (uppercase or lowercase)**. 5. It must contain **at least one special character** from the above set. Use **four separate regex patterns** and combine them using logical **and**, similar to the example shown below. After defining the function, test it using the following inputs and print whether each password is valid (True) or invalid (False):	L2, L3

Q3.	Write normalize_phones(s) that takes a string with phone numbers in inconsistent formats and returns a string where all numbers are normalized to +91-XXXXXXXXXX. Accept examples like +91-9876543210, (91) 98765 43210, 0091 9876543210, 91 9876543210. Use character sets, \s, optional groups (?) and quantifiers — do **not** use numeric-range parsing beyond regex. **Input:** "Contact: +91-9876543210, Office: (91) 98765 43210, Home: 0091 9876543210" **Output:** "Contact: +91-9876543210, Office: +91-9876543210, Home: +91-9876543210"	L2, L3
Q4.	Using re.search, find the **first** substring of two-or-more digits in a target string and print the matched text and its span (start,end) using the match object. **Example Input:** "1 set of 23 owls, 999 doves." **Expected output printed:** "23" found at (9, 11) (Use pattern \d{2,} and m.group() + m.span()).	L1, L2
Q5.	Write a function that, given a word, returns True if it matches the pattern: **starts with c, then a vowel (a,e,i,o,u), then any single consonant**, using appropriate character sets and ranges. For example cat, cit, cot, cut should match; caa should not (because third char must be consonant).	L1, L2
Q6.	Write a regex that matches any string that **begins** with one or more lowercase letters (`a-z`), followed by one or more digits (`0-9`), and nothing else (`^...$`). Provide a short Python snippet to test several example strings and list which pass. **Examples:** `abc123` → pass, `a1b2` → fail, `ABC123` → fail (uppercase not allowed by pattern).	L1, L2
Q7.	Given the text: "Random <tag>first</tag> some text <tag>second</tag> end" (a) Write a regex using a quantifier that will match from the first <tag> to the final </tag> (show what is matched). (b) Then write a regex using a quantifier so you capture each tag pair separately (show both matches).	L2, L3
Q8.	Write `extract_date_parts(s)` that uses a regex with **three groups** to parse a date of form `DD-MM-YYYY` (digits). If found, return a tuple: `(whole_match, day, month, year, span_of_whole_match, lastindex)`. Use `re.search` and the match object's methods (`group(0)`, `group(1)`, `group(2)`, `group(3)`, `span()`, `lastindex`) to build the tuple.	L2, L3

	Example: input `"Backup 05-11-2025 complete"` → output `('05-11-2025', '05', '11', '2025', (7,17), 3)`	
Q9.	Write a small function that uses `re.search` to locate the **first** substring that is an uppercase letter followed somewhere after it by a digit (the digit does not need to be immediately adjacent). Return the matched substring of the uppercase letter (i.e., `group(0)` or appropriate capture) and position. Use a concise pattern that uses `.*` and capture the uppercase letter as a group. **Example:** `"a B blah 9"` → match uppercase `B`, and its index. .	L2, L3

Q10.	Write a Python function `find_repeated_letters(s)` that finds **all occurrences** where **the same letter appears twice or more in a row** (like `aa`, `bbb`, `zzzz`).
	Use:
	
	• a **capturing group** for the letter: `(.)`
	• a quantifier referencing the same letter: `\1+`
	• `re.finditer()` to retrieve **all** match objects
	• For each match object, print:
	  ○ the repeated sequence (`group(0)`)
	  ○ the repeated character (`group(1)`)
	  ○ the span (`span()`)
		L2, L3

	-END-	