Faculty of Technology and Engineering

Date: 01/07/ 2025

Academic Year	:	2025-2026	Semester	:	1
Course code	:	CEUC101	Course name	:	Computer Concepts and Programming

Sr. No.	Practical Definition	Hrs	CO
	Basic create and run C program		
1.	In a bustling computer science department, a group of curious students gathered around their professor, eager to learn the ins and outs of programming. Dr. Techwise, their ever-inventive instructor, decided to give them a practical challenge that would bridge the gap between theory and application. "Imagine," Dr. Techwise began, "that you are explorers venturing into the world of programming tools and environments. Your mission is to write a simple C program that displays your name on the screen. But here's the twist—you must do this using different text editors and Integrated Development Environments (IDEs) across Windows and Linux systems. Each environment is a unique terrain with its own set of tools and files. Ready to embark on this adventure?" Try to learn the backend files formed between the source code file and output file. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. What are the steps you followed to write, compile, and execute the C program? 2. What editor or IDE did you use on Windows and Linux respectively? 3. What file extensions were created at each stage (e.g., .c, .o, .exe, .out)? **Supplementary Problems -** 1. Rename the output file during compilation. 2. Identify and list all intermediate and final files created. 3. Compare: Compilation command used in GCC on Linux vs. TDM-GCC or MinGW on Windows 4. Compare: Behavior of program output across platforms **Key Skills to be addressed – Conceptual Skills (**Understanding the C compilation process, File lifecycle: source → object → executable, etc.), Technical Skills, Problem-Solving Skills **Applications –** Software Development Pipelines, Cross-Platform Compatibility **Learning Outcome -** Students will understand how source code becomes executable across platforms, explore compilation process, file types generated, and the role of different text editors and IDEs.	2	CO1
	Tools/Technology to Be Used: Linux: gedit, gcc, Windows: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours		

	* Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
	Data Types, Variables, Input, Output		
2.	Mr. Compute, the head of the planning department, gathered the team and laid out a map of the proposed park. He pointed to the dimensions of the rectangular plot, elegantly sketched with labeled measurements: Length: A grand 70 meters, stretching long enough to host a jogging track. Breadth: A modest but substantial 90 meters, providing ample space for a variety of activities. "These dimensions," Mr. Compute explained, "make this park a perfect fit for our community's needs. But we must calculate the area and perimeter to finalize the blueprint." **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Did the student use appropriate data types and variable names? 2. Does the program handle different inputs? 3. Is the output formatted clearly? **Supplementary Problems -** • Add functionality to calculate **fencing cost** and **grass laying cost** based on given rates. • Modify the program to calculate the area and perimeter of **multiple parks** using loops. • Add input validation to ensure dimensions are positive. **Key Skills to be addressed –** Mathematical Reasoning, Problem Solving, Analytical Thinking, Basic Programming **Applications –** Programming **Learning Outcome -** Ability to translate real-life problems into computational logic. Understanding how programming can support planning, design, and engineering tasks. Use of arithmetic and input/output in C.	1	CO1
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
3.	In the heart of a bustling city, a popular fitness center, *ActiveLife Studio*, was thriving with members who were determined to improve their health and well-being. The studio's manager, Priya, was passionate about helping people achieve their fitness goals. One day, during a meeting with her trainers, Priya raised an important point: "Our members often ask if they're making progress, but they don't have an easy way to track their overall health. What if we introduce a **Body Mass Index (BMI)** calculator as part of our program? It's simple, yet powerful for monitoring health." The team was excited about the idea, but there was a challenge: they needed a reliable and efficient tool to calculate BMI for all members.	1	CO1

	Priya turned to a tech-savvy friend, Arjun, for help. She explained the requirements: 1. The program must accept a person's **weight in kilograms** and **height in meters** as inputs. 2. It should calculate the BMI using the formula: $BMI=Weight/Height^2$ 3. Finally, the program should display the BMI in a clear, user-friendly format, so members could easily understand their results. "This will not only empower our members to monitor their health," Priya added, "but it will also encourage them to adopt healthier habits!" **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the BMI calculated correctly using the given formula? 2. Are the inputs (weight and height) validated (non-zero, positive)? 3. Does the program display BMI with appropriate formatting (e.g., 2 decimal places)? 4. Can the user easily interpret the result? Is there any categorization (e.g., underweight, normal, overweight)? **Supplementary Problems -** 1. Add functionality to calculate **fencing cost** and **grass laying cost** based on given rates. 2. Modify the program to calculate the area and perimeter of **multiple parks** using loops. 3. Add input validation to ensure dimensions are positive. **Key Skills to be addressed –** Mathematical Reasoning, Problem Solving, Analytical Thinking, Basic Programming **Applications –** Programming **Learning Outcome -** Ability to translate real-life problems into computational logic. Understanding how programming can support planning, design, and engineering tasks. Use of arithmetic and input/output in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
4.	At *TechNest University*, the programming club *CodeMasters* decided to up the ante for their annual workshop. After the success of their data type demonstration last year, they wanted to make this year's event even more interactive and insightful for new students. During a brainstorming session, the club's president, Anya, exclaimed, "Let's show them the magic of data types **and** their ranges! If students understand not just the size but also the capabilities of each data type, they'll be able to write better programs with precision and confidence." The team cheered in agreement, and a new challenge was born.	1	CO1

	• Discover **the Data Types:** Write a program that displays the **size (in bytes)** of commonly used data types (`char`, `int`, `float`, `double`, etc.) in C. • Explore **the Ranges:** Include a feature that outputs the **minimum and maximum values** of each data type using appropriate constants from `<limits.h>` and `<float.h>`. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. What is the significance of using different data types in C? 2. How does memory size vary across data types? 3. Are <limits.h> and <float.h> appropriately used to fetch min/max values? 4. What are the implications of signed vs unsigned types? **Supplementary Problems -** 1. Ask the user to enter a number and validate if it fits within a chosen data type's range. 2. Display sizes and ranges for short, long, long long, unsigned types.. 3. Explore size_t, ptrdiff_t, and their applications. **Key Skills to be addressed –** Memory management awareness, Precision and overflow handling, Practical C programming (input/output, variables, headers) **Applications –** Writing efficient and optimized code. **Learning Outcome -** Ability to identify and describe the size and range of basic data types in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
5.	In the bustling offices of Bharat's Home Ministry, a team of data analysts sat in a meeting room, surrounded by charts, reports, and coffee cups. The year was 2024, and the Ministry had embarked on a critical mission: to better understand the state of education in Bharat. Minister Rajan, known for his forward-thinking strategies, addressed the room: "We are on a journey to improve literacy in Bharat. To make informed decisions, we need accurate data on the number of literate and illiterate men and women in our population. Let's turn these numbers into actionable insights." The analysts were given the following statistics: 1. **Population of Bharat in 2024:** 1,441,981,744 2. **Percentage of Women:** 48.4% 3. **Literacy Rates:** o **Overall:** 85.95% o **Men:** 80.95% o **Women:** 62.84% The team's challenge was to compute the count of **illiterate men and women** using these statistics. To streamline this process, they turned to their tech-savvy intern, Arya.	1	CO1

	Key Questions / Analysis / Interpretation to be evaluated during/after Implementation 1. How do we calculate the number of men and women from the total population? 2. How do we ensure the correct use of float and long data types for large numbers? 3. How is literacy calculated using percentage? **Supplementary Problems -** 1. Calculate literate population (men and women separately, and in total). 2. Compare literacy gap between genders and output a message accordingly. 3. Generate a simple **tabular report** using formatted output (printf with alignment).. 4. Add a feature to accept population and percentages as **user input** (instead of fixed values). 5. **What-if Analysis**: If the literacy rate improves by 5% for women, what would be the new number of literate and illiterate women? **Key Skills to be addressed –** Arithmetic & Percentage, Problem Decomposition **Applications –** Real-time problem solving. **Learning Outcome -** Ability to apply mathematical formulas in a programming context.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
6.	In the heart of India's bustling Meteorological Department, meteorologist Riya stared at the glowing screen filled with weather data. It was a critical day – the team was preparing a comprehensive weather report for an upcoming international climate summit. Suddenly, her colleague Arjun exclaimed, "Riya, we need the temperature readings in Fahrenheit for our global audience. Most countries use Fahrenheit for their weather updates!" Riya nodded. While the team's sensors recorded temperatures in Celsius, the global report required conversions to Fahrenheit. "Let's automate this," Riya suggested. "We need a program to do the conversion accurately and quickly." $Fahrenheit = \left(Celsius \times \frac{9}{5}\right) + 32.$ Analyze the value of Fahrenheit while temperature is 0, 100, or -40 Celsius. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Do students correctly apply the formula: Fahrenheit = (Celsius × 9/5) + 32? 2. Is the division performed in **floating-point** to ensure accuracy?	1	CO1

	3. What happens when Celsius = 0, 100, or -40? **Supplementary Problems -** 1. Extend the program to allow conversion **both ways** (Celsius ⇄ Fahrenheit) using a menu 2. Add conversion options to/from **Kelvin**, expanding understanding of temperature systems.. 3. Convert a **range of Celsius values** (e.g., -20 to 100) and display a table. 4. Visualize temperature ranges using symbols (e.g., bars for hot/cold levels). **Key Skills to be addressed –** Arithmetic expressions and handling floating-point precision **Applications –** Weather reporting software and dashboards. **Learning Outcome -** Ability to recognize the importance of **data types**, especially for precision in real-world measurements.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
	Branch Statements		
7.	At the Sunrise Amusement Park, the management is introducing an automated ticketing system to reduce wait times at entry gates. The system needs to decide whether a visitor should be charged for entry or allowed in for free. The park allows free entry to children but charges a fixed ticket price for adults. The management has provided some basic guidelines for implementing this system: • Visitors will provide their age at the entry gate. • Depending on their age, the system will decide whether the visitor is eligible for free entry or must pay for a ticket. **Note:** Programmers have rights to choose the constant ticket fare for adults and workout accordingly. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. What age is considered the cut-off for free entry? (e.g., <12 years) 2. How is the ticket fare stored and applied in the program? 3. Can the program distinguish clearly between children and adults? 4. How can the solution scale to handle multiple entries or pricing tiers (e.g., senior citizens)? **Supplementary Problems -** 1. Add another condition to offer a discount or free entry to senior citizens (e.g., age ≥ 60). 2. If the number of adults in a group exceeds 5, apply a 10% discount on the total. 3. Ensure age is within a realistic range (e.g., 0–120). Show an error message otherwise.	1	CO2

	4. Add a menu allowing users to select visitor type: Child, Adult, Senior, and show ticket cost accordingly. **Key Skills to be addressed –** Decision-making, logical reasoning **Applications –** Smart entry gates in events, museums, and parks. **Learning Outcome -** Ability to understand and implement **conditional logic** using if, else if, and else statements.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
8.	Spark Savings Bank is launching a new campaign to attract young adults to open savings accounts. To ensure only eligible individuals apply, the bank wants to introduce an eligibility checker at their branches. Here's the scenario: 1. Customers walk into the branch and provide their age. 2. Depending on their age, they are either deemed eligible to open a savings account or politely informed that they are not eligible yet. The eligibility criteria and how the system should behave are left for you to deduce and implement. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. What is the **minimum eligible age** to open a savings account (Commonly assumed as **18 years**)? 2. Is the program handling **invalid input** like negative numbers or characters? 3. Are conditions clearly structured using if-else or switch-case/ else-if ladder? 4. Is the output **clear, polite, and user-friendly**? **Supplementary Problems -** 1. Multi-tier Eligibility: Classify customers: a. Below 18: Not eligible b. 18–59: Eligible for Regular Savings c. 60 and above: Eligible for Senior Citizen Account 2. Add a menu: 1. Check eligibility, 2. Exit. Include a loop to allow repeated checks.. 3. Accept and evaluate age for **multiple customers** using a loop or array. **Key Skills to be addressed –** Conditional logic (if-else), Scalability and real-world adaptability **Applications –** Customer verification in banks and service centers **Learning Outcome -** Ability to translate real-world eligibility rules into working logic in C.	1	
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		

	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
9.	Every year during the festive season, **ShopEase Online** introduces special discounts to attract customers and boost sales. This year, the store's marketing team has come up with a tiered discount policy to reward loyal shoppers based on their spending. As part of their customer engagement drive, they've tasked you, a budding developer, with designing the logic for the discount system. Here's the scenario: 1. Customers shop for their favorite items and proceed to the checkout page. 2. The system calculates the total shopping amount and applies a discount based on their spending: o For total amounts below ₹1000, no discount is applied. o For totals between ₹1000 and ₹5000, the customer enjoys a **10% discount**. o For totals exceeding ₹5000, they receive a generous **20% discount**. 3. The system must display the total amount, the discount applied, and the final amount the customer needs to pay. The store manager is eager to see how this system can enhance customer satisfaction and drive sales. How will you design a program that ensures every customer experiences a seamless checkout process with accurate discounts? **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the program applying the correct discount tier based on the input? 2. Is the calculation for discount and final payable amount accurate?? 3. Is the output user-friendly and informative (i.e., all values displayed clearly)? 4. What happens when the user enters boundary values like exactly ₹1000 or ₹5000? **Supplementary Problems -** 1. After discount, apply 5% GST on the final amount. 2. Add an extra 5% discount if the customer is a registered member. 3. Accept a promo code input and apply additional discounts if valid. 4. Use a loop to process multiple customers' bills in one run. **Key Skills to be addressed –** Logical reasoning, decision-making, arithmetic logic **Applications –** E-commerce platforms **Learning Outcome -** Ability to handle real-world scenarios involving tiered pricing or decision-making in C.	1	CO2
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
10.	Foodies is a restaurant established in 2020. Due to rush of customers, the waiters are not able to manage the food supply appropriately. To help them, create a menu ordering system, which allows customers to select the items from the menu and	1	CO2

	compute the total cost of their order. Use switch case for menu ordering and item selection. Menu includes the following: • Burger - ₹150 • Pizza - ₹200 • Pasta - ₹120 • Sandwich - ₹100 • French Fries - ₹80 Display the menu to user and allow them to enter the number they wish to order. Calculate the total amount after selection of all items. Ask user to enter '0' after finishing the ordering of items. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the menu presented clearly with numbered options and prices? Does the user receive clear instructions for ordering and exiting (Enter 0 to finish)? 2. Is each menu item correctly mapped to its price in the switch-case block? 3. Is the total displayed accurately once ordering is complete? 4. Is input case handled properly (e.g., rejecting negative inputs or non-menu values)? **Supplementary Problems -** 1. Allow users to enter the **quantity** of each item ordered and multiply cost accordingly. 2. Use a loop to allow ordering of multiple items in one session. 3. Apply a **discount** if the total exceeds ₹500. 4. Display a summary of all items ordered with quantity and individual totals. 5. Implement item names and prices using arrays or struct for better scalability. **Key Skills to be addressed –** Use of **switch-case** control structure **Applications –** Food delivery app backend logic **Learning Outcome -** Ability to use modular code to handle repetitive user interactions in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
11.	Determine the grade of a student based on their marks using the conditional (ternary) operator. Take student's marks as input and display the corresponding grade as output according to the following criteria: • Marks >= 90: Grade A • Marks >= 80 and < 90: Grade B • Marks >= 70 and < 80: Grade C • Marks >= 60 and < 70: Grade D • Marks < 60: Grade F Validate the input by ensuring user is entering marks between 0-100, else declare that the entered input is invalid. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation**	1	CO2

	1. Is the ternary operator used correctly and efficiently for multiple conditions? 2. Is the **input validated** to ensure marks are within the 0–100 range?? 3. Does the output correctly reflect the corresponding grade based on the input? 4. How does the logic behave at **boundary conditions** (e.g., 89.9, 90, 59.9)? **Supplementary Problems -** 1. Extend to GPA: Assign numeric GPA instead of grades (e.g., A=4.0, B=3.0…). 2. Compare with if-else: Implement the same logic using if-else and discuss readability. 3. Grade with Remarks: Add remarks along with grades (e.g., "Excellent", "Needs Improvement"). 4. Multiple Students: Use a loop to take grades for multiple students and display a summary. 5. Input in Float: Accept marks in decimal (e.g., 89.5) and apply logic accordingly. **Key Skills to be addressed –** Decision-making, validation, classification, Ternary operator usage **Applications –** Automating grade calculations **Learning Outcome -** Ability to understand the syntax and use of the **ternary operator** in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
Looping Statements			
12.	The librarian at City High School has implemented a barcode system for all library books. Each book is assigned a unique ID number from 1 to 50. The librarian wants to generate a catalog to: Display all the book IDs sequentially. Mark every 5th book as a "Special Edition". Write a program to generate this catalog, ensuring it displays each book ID on a new line and highlights special editions with a clear note next to their ID. **Expected sample outcome:** **Book ID: 1** **Book ID: 2** **Book ID: 3** **Book ID: 4** **Book ID: 5 (Special Edition)** **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation**	2	CO2

	1. Is a **loop structure** (for, while, etc.) used efficiently to iterate from **1 to 50?** 2. Is the condition for checking **every 5th book (book ID % 5 == 0)** correctly applied? 3. Is each book ID printed on a new line? Are "Special Edition" books clearly marked and formatted for readability? 4. Can the same logic be extended easily if the range changes (e.g., 1 to 100)? **Supplementary Problems -** 1. Allow the user to enter the **start and end** range for book IDs 2. Let the user input which books to mark as special (e.g., every 3rd or 7th book). 3. Extend logic to allow **multiple categories** (e.g., every 5th = "Special", every 10th = "Rare"). 4. Write the catalog to a **text file** instead of just printing to the screen. **Key Skills to be addressed –** Looping structures (for, while) **Applications –** Book or inventory cataloging systems **Learning Outcome -** Ability to understand how to **apply loop and conditional logic** to solve real-world problems in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
13.	The **Smart Water System Company** has developed an automatic tank refill system. This system starts refilling an empty tank (initially at 0 liters) until it reaches its maximum capacity of **100 liters**. • The system refills the tank at a fixed rate of **10 liters per minute**. • It displays the current water level after each refill. Write a program to simulate this process. Ensure the program stops automatically when the tank is full, displaying a message: **"Tank is full."** **Expected sample outcome:** **Current water level: 10 liters** **Current water level: 20 liters** **...** **Tank is full.** **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the loop correctly implemented to simulate time-based refill? 2. Does the **output update accurately** in increments of 10 liters? 3. Does the message "Tank is full." display only once after reaching capacity? 4. Is the **termination condition** (tank == 100) handled properly? **Supplementary Problems -** 1. Accept refill rate and tank capacity as input and adjust logic accordingly. 2. Add a delay (e.g., sleep(1) second) between updates to simulate real-time behavior. 3. Print water level as a percentage: e.g., "Tank is 60% full".	1	CO2

	Key Skills to be addressed – Loop reasoning, sequential logic **Applications –** Smart home automation **Learning Outcome -** Ability to understand and implement **looping constructs** effectively in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
14.	The National Bank of Bharat wants to simulate an ATM machine. Assume the user has a starting balance of ₹5000. The ATM should allow the user to withdraw cash as long as their balance is sufficient. For every transaction: • Prompt the user to enter the amount they wish to withdraw. • Deduct the amount from the balance and display the remaining balance. • If the user attempts to withdraw more than the available balance, display a warning message: "Insufficient balance." The program should continue until the user chooses to stop. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is there a proper mechanism (e.g., menu or confirmation) for the user to **exit the loop**? 2. Is the withdrawal amount correctly deducted from the balance? Is the "Insufficient balance" message shown accurately when needed?? 3. Does the program handle: Negative amounts, Zero withdrawal and Non-numeric inputs (optional challenge)? 4. Are **current and remaining balances** displayed clearly? **Supplementary Problems -** 1. Allow the user to **deposit** money in addition to withdrawal. 2. Prompt the user for a **PIN number** before allowing transactions.. 3. Limit the number of daily withdrawals (e.g., max 5 times per session). 4. Save transaction details to a **text file** as a mini-statement. **Key Skills to be addressed – Looping structures** (while, do-while), **User-driven control flow** using menu or flag variables **Applications –** Simulation of real-world banking systems **Learning Outcome -** Ability to develop interactive, **user-controlled programs** in C.	1	
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
15.	Develop a countdown timer, that allows user to set a starting number of seconds and then count down to zero, displaying each second as it decrements. After the countdown completes write 'Countdown completed!'.	1	CO2

	Key Questions / Analysis / Interpretation to be evaluated during/after Implementation 1. Is the user input being validated to ensure a non-negative value? 2. Is the countdown displayed correctly in decrementing order? 3. Does the loop terminate correctly when zero is reached? 4. Is the message "Countdown completed!" displayed at the end only once? **Supplementary Problems -** 1. Use sleep(1) or a platform-specific delay function to simulate real-time countdown. 2. Ask the user if they want to restart another countdown. 3. Format the output in minutes and seconds (e.g., 01:30, 01:29, ...). **Key Skills to be addressed – Sequencing, timing logic, reverse iteration** **Applications – Alarm systems** or safety shutdown mechanisms **Learning Outcome -** Ability to implement **countdown logic** using loops in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
16.	Develop a C program that simulates a matchstick game between the user and the computer. The objective of the game is to avoid picking the last matchstick. The program should ensure that the computer always wins by strategically picking matchsticks. The game starts with 21 matchsticks. The user and the computer take turns to pick 1, 2, 3, or 4 matchsticks. The player forced to pick the last matchstick loses the game. **Rules**: 1. The game starts with 21 matchsticks. 2. The user is asked to pick 1, 2, 3, or 4 matchsticks. 3. After the user picks, the computer makes its pick. 4. The player who is forced to pick the last matchstick loses the game **To understand the above game in a better way, visit the following link:** http://atozmath.com/Games/21MatchStick.aspx **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Do students understand that the computer should always leave a multiple of 5 matchsticks after its turn? Is the computer's move calculated as: computer_pick = 5 - user_pick;? 2. Does the program ensure the user picks only **1 to 4 matchsticks**? 3. Does the program detect when the game is over and **declare the winner appropriately**? **Supplementary Problems -** 1. Add an option for the user to **choose who starts first**.. 2. Modify the computer's logic to **play randomly**, and compare win/loss patterns. 3. Allow the user to **play again** after a match is completed. 4. Track and display the **user and computer win count** across multiple games.	1	

	Key Skills to be addressed – Looping structures (while, do-while), **Mathematical reasoning (strategy implementation), Strategic programming with predictable outcomes** **Applications –** Developing **turn-based strategy games** **Learning Outcome -** Ability to learn how to apply **game theory and mathematical strategy** in C programming.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
17.	The National Sports Federation has developed an app to track a runner's distance during a marathon. The runner starts from 0 km and covers 0.5 km every minute. The app needs to display the distance covered at each minute until the runner completes a total distance of 10 km. Write a program to simulate the distance tracking. **Note:** Use an infinite while loop with a break statement to stop tracking after the runner has completed the marathon. **Expected sample outcome:** **Minute 1: Distance covered = 0.5 km** **Minute 2: Distance covered = 1.0 km** **...** **Minute 20: Distance covered = 10.0 km** **Marathon complete!** **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the infinite loop properly controlled using a break condition? 2. Are **minutes** and **distance** tracked and displayed correctly? 3. What happens if we change the **step size** or total **distance goal**? 4. Is the use of float or double appropriate for distance calculations? **Supplementary Problems -** 1. Let the user input the marathon distance (e.g., 5 km, 15 km). 2. Allow users to enter the distance covered per minute dynamically. 3. At the end, print total minutes taken. 4. Add a delay (e.g., `sleep(1)`) to mimic real-time tracking. **Key Skills to be addressed – Pattern recognition, condition monitoring, Controlled exit from infinite loops** **Applications –** Fitness tracking applications (e.g., running, walking, cycling) **Learning Outcome -** Ability to understand the usage and control of **infinite loops** using while(1) and break in C.	1	CO2
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		

18.	Write a C program to generate and display a multiplication table based on user input in the following form:
	`Enter the size of table vertically: 10`
	`Enter the size of table horizontally: 5`
	
	`Multiplication Table (10 x 5):`
	`1 2 3 4 5`
	`2 4 6 8 10`
	`3 6 9 12 15`
	`4 8 12 16 20`
	`5 10 15 20 25`
	`6 12 18 24 30`
	`7 14 21 28 35`
	`8 16 24 32 40`
	`9 18 27 36 45`
	`10 20 30 40 50`
	
	Key Questions / Analysis / Interpretation to be evaluated during/after Implementation
	1. Are **invalid inputs** (e.g., negative numbers, zero) properly handled?
	2. Is the multiplication table **neatly formatted and readable**?
	3. Can the program be easily modified to print tables for multiple numbers?
	Supplementary Problems -
	1. Allow the user to specify how far the table should go (e.g., up to 20 or 50).
	2. Display the table in **reverse order** (e.g., 10 × n down to 1 × n).
	3. Write the multiplication table to a **text file**.
	Key Skills to be addressed – Looping structures (while, do-while)
	
	Applications – Educational software (e.g., teaching basic math)
	Learning Outcome - Ability to Perform arithmetic operations and display results using **formatted output in** C.
		1	CO2

	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
19.	Use appropriate nested loops to draw the following patterns:	4	CO2
	(i) 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1		
	(ii) 1 2 3 4 5 6 7 8 9 2 8 3 7 4 6 5		
	(iii) 5 4 3 2 1 2 3 4 5 4 3 2 1 2 3 4 3 2 1 2 3 2 1 2 1 2 1 2 3 2 1 2 3 4 3 2 1 2 3 4 5 4 3 2 1 2 3 4 5		
	(iv) A ABA ABCBA ABCDCBA ABCBA ABA A		
	Key Questions / Analysis / Interpretation to be evaluated during/after Implementation 1. Are **nested loops** correctly structured (outer for rows, inner for columns)? 2. Is the **spacing logic** accurate for alignment (especially for pyramids)?		

	3. Is the program flexible (can it handle variable input sizes)? 4. How do **loop counters** control pattern size and shape? 5. Can the student identify the **row-column relationship**? **Supplementary Problems -** 1. Develop a **Hollow Square** 2. **Pascal's Triangle** (Advanced) 1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 1 5 10 10 5 1 1 6 15 20 15 6 1 1 7 21 35 35 21 7 1 3. **Butterfly Pattern**  **Key Skills to be addressed – Nested loops (for, while), conditional logic** **Applications – Console-based UI design** (e.g., progress bars, loading visuals) **Learning Outcome -** Ability to apply **row-column logic** to build structured outputs in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 3 hours * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
	Arrays		
20.	You are a security officer responsible for tracking attendance at an exclusive workshop with N participants. At the end of the event, you have a list of N-1 unique participant IDs representing those who attended. However, one participant ID is missing from the list. Your task is to identify the missing participant ID from the range of IDs, which goes sequentially from 1 to N, to ensure accurate records. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Do students understand that participant IDs range **sequentially from 1 to N**, and one ID is missing in the provided list of **N–1 unique IDs**? 2. Is the list **unordered** or **sorted**? Does the method account for both? 3. Which method is used to find the missing ID:? 4. Are all IDs read correctly using loops and arrays? 5. What happens if **no ID is missing** or if **multiple IDs** are missing?	2	CO3

	Supplementary Problems - 1. Modify the program to detect **more than one missing ID** from the range. 2. Detect if any participant ID is **repeated**, violating the "unique" assumption. 3. Accept and process **unsorted participant IDs** (realistic scenario). 4. Read the list of IDs from a **text file** for larger test cases. **Key Skills to be addressed –** Loops and arrays, Arithmetic and logic operations **Applications –** Attendance tracking in events, workshops, or classrooms **Learning Outcome -** Ability to strengthen skills in input handling, array processing, and data validation in C Programming.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
21.	In many real-world settings, such as government offices, banks, or survey agencies, **data entry operators** often input numerical data into systems manually. To aid in **quick analysis and validation of input data**, you are tasked with developing a **C-based console application** that assists operators in understanding the nature of the numbers they enter. The system should allow the operator to input exactly **25 integers** through the keyboard. Once all the numbers are entered, the program should process the data and display the following statistics: • Total number of **positive** numbers • Total number of **negative** numbers • Total number of **even** numbers • Total number of **odd** numbers This program will help data entry staff and analysts to **quickly interpret trends**, check for errors (e.g., unexpected negative values), and prepare data for further processing or reporting. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the program correctly reading exactly 25 integers from the user? 2. Is the logic for identifying positive/negative and even/odd numbers implemented accurately? 3. Are separate counters used for each category (positive, negative, even, odd)? 4. Are the final counts for each category displayed clearly to the user? **Supplementary Problems -** 1. Allow the user to enter the number of values (n) instead of fixing it to 25. 2. Count and report the number of **zeros** separately. 3. Store positive, negative, even, and odd numbers in **separate arrays**, and display them. 4. Read numbers from a file and write results to a report file. **Key Skills to be addressed –** Array traversal and indexing, **Loops** and **conditional statements** in C **Applications –** Quick statistical analysis of raw data inputs **Learning Outcome -** Ability to implement a basic **statistical computation system** using C..	1	CO3

	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		

22.	Display the seating arrangement in theatre using C program. The theater has a fixed number of rows and seats per row. Create a seating chart where each seat is identified by its row and seat number. Additionally, the program should allow the user to mark certain seats as reserved. The seating chart should be displayed with indicators showing which seats are reserved and which are available.
	Expected Outcome:
	`Enter the number of reserved seats: 3`
	`Enter row and seat number for reserved seat 1 (e.g., 2 5): 1 3`
	`Enter row and seat number for reserved seat 2 (e.g., 2 5): 2 5`
	`Enter row and seat number for reserved seat 3 (e.g., 2 5): 3 7`
	
	`Seating Chart:`
	`Row 1: O O X O O O O O O O`
	`Row 2: O O O O X O O O O O`
	`Row 3: O O O O O O X O O O`
	`Row 4: O O O O O O O O O O`
	`Row 5: O O O O O O O O O O`
	
	Key Questions / Analysis / Interpretation to be evaluated during/after Implementation
	1. How are seats represented? Is a **2D array** used to model rows and columns (seats)?
	2. Can the user **select specific seats** to mark as reserved?
	3. Does the program gracefully handle **invalid or out-of-bound input**?
	4. Are reserved and available seats **clearly marked** (e.g., 'R' for Reserved, 'A' for Available)?
	5. Is the seating chart displayed in a **readable format**?
	Supplementary Problems -
	1. Allow the user to define the **number of rows and seats per row** at runtime.
	2. Display the **number of seats reserved and available**.
	3. Let users check if a specific seat is available before booking.
	4. Add the ability to cancel a reservation.
	5. Allow booking of **consecutive seats** for groups or families.
	Key Skills to be addressed – 2D array manipulation
	
	Applications – Real-life **ticket booking systems** (theaters, events, transport)
	Learning Outcome - Ability to understand how to model **grid-based systems** using arrays in C.
		1	CO3

	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 40 mins * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		

23.	You are managing a fruit stand at a local market. The market operates for several days, and you record the price of a specific fruit every day in an array, where `prices[i]` is the price of the fruit on the `i-th` day. As a vendor, your goal is to maximize your profit by buying the fruit on one day and selling it on a later day when the price is higher. You can make only one transaction (buy one day, sell one day). Write a program in C to determine the maximum profit you can achieve from this transaction. If no profit is possible (i.e., the fruit price only decreases or stays the same), return `0`. **Test cases:**	2	CO3

Test Case	Input (`prices`)	Output (Profit)	Explanation
1	`[20, 25, 15, 30, 10, 50]`	`40`	Buy on day 5 (price = 10) and sell on day 6 (price = 50). Profit = `50 - 10 = 40`.
2	`[10, 8, 6, 4, 2]`	`0`	No transaction is possible as the price keeps decreasing. Maximum profit = `0`.
3	`[100, 180, 260, 310, 40, 535, 695]`	`?`	
4	`[30, 20, 25, 40, 25, 50, 35]`	`?`	
5	`[5, 5, 5, 5, 5]`	`?`	

Key Questions / Analysis / Interpretation to be evaluated during/after Implementation

1. Do they understand the **importance of order** (i.e., sell must come after buy)?
2. Are **minimum price tracking** and **profit calculation** done correctly?
3. Does the code correctly return 0 when no profit is possible (e.g., prices only decrease)?

Supplementary Problems -

1. Modify the program to calculate the **maximum total profit** from **multiple buy/sell pairs**.
2. Output the days (indices) of buying and selling for max profit.
3. Store prices and day numbers in a struct and print detailed transactions.
4. Read the price list from a file and write the result to an output file.

Key Skills to be addressed – Array traversal and indexing

Applications – Retail profit analysis (when to buy and sell stock/inventory)
Learning Outcome - Ability to analyze problems from both **logical and mathematical** perspectives and implement it in C.

Tools/Technology to Be Used: Code:blocks		
* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hour		
Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		

24.	Alex is a budding writer who loves jotting down creative ideas and thoughts. However, Alex prefers using simple, lightweight tools over complex software. Frustrated by the clutter of unnecessary features in modern note-taking apps, Alex decides to create a custom text-based note-taking program tailored to their needs. As a developer, you are tasked with helping Alex design this application. The program should allow Alex to manage notes and perform the following operations manually (without using built-in string manipulation functions from <string.h>): 1. Calculate the Length: Alex wants to know the length of a specific note to meet character count requirements. 2. Reverse a Note: Alex often enjoys viewing their notes in reverse order as a creative exercise. 3. Compare Two Notes: Sometimes, Alex drafts multiple versions of the same idea and needs to compare them to identify similarities or differences. 4. Copy a Note: Alex frequently copies notes to create variations or backup content. 5. Concatenate Notes: When merging ideas, Alex needs to combine two notes into one seamlessly. 6. Upper Case Converter: often tries to convert whole note into upper case. 7. Lower Case Converter: often tries to convert whole note into lower case. 8. Capitalize each word: In whole note, tries to capitalize 1st character of each word. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Are all operations implemented using **manual character-wise logic**? 2. Is the **menu interface user-friendly** and functional? 3. Are **edge cases** (e.g., empty strings, large notes, unequal lengths) handled? 4. Is memory usage and note size management done effectively? **Supplementary Problems -** 1. Implement a **"note history"** to track last few operations. 2. Implement Undo Operation. 3. Save and load notes from a file using **basic file I/O**. **Key Skills to be addressed –** Manual string manipulation in C **Applications –** Core of **text editing software** or **terminal-based utilities** **Learning Outcome -** Ability to gain a deep understanding of how strings work internally in C.	4	CO3
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 3 hours * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
	User Defined Functions		
25.	A library manager wants to build a simple system to manage and track book-related tasks such as checking book availability, updating inventory, and calculating fines for overdue books. Different methods are required to address these operations. Expected Outcome:	4	CO4

Method Type	Scenario	Function Signature	Input	Output
No arguments, no return value	Display list of available books	`void displayBooks()`	None	Prints list of books
No arguments, return value	Get total number of books	`int getTotalBooks()`	None	Returns total count
Argument passed, no return value	Borrow a book	`void borrowBook(char* bookName)`	Book name	Updates inventory, prints
Argument passed, return value	Calculate fine for overdue book	`float calculateFine(int daysLate)`	Days overdue	Returns fine amount

Key Questions / Analysis / Interpretation to be evaluated during/after Implementation

1. Are function **signatures properly defined** based on the task (return type and parameters)?
2. Are book records stored using appropriate data structures (e.g., arrays of strings, structs)?
3. Are return values used meaningfully (e.g., int for total books, float for fines)?
4. Are inputs such as book names or days late **validated** to avoid errors or incorrect calculations?
5. Is there a clear **difference between print-only functions and value-returning functions**?
6. Is the **book availability check** implemented before allowing borrowing? Does borrowing a book correctly **update the available book count or mark it as unavailable**?
7. Can these functions be reused in other library scenarios (e.g., returning books, viewing borrowed books)?

Supplementary Problems -

1. Implement void returnBook(char* bookName) to update availability.
2. Implement int searchBook(char* bookName) returning 1 if found, 0 otherwise.
3. Store due dates and create a function to list all overdue books.
4. Associate books with user IDs and update borrowing/return logs.
5. Save and retrieve book lists and inventory from text files using fopen(), fprintf(), etc.

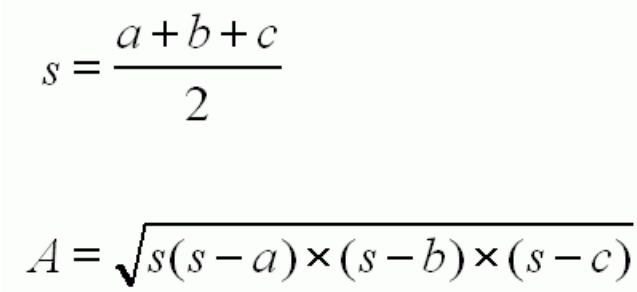
Key Skills to be addressed – Function declaration, definition, and usage in C. Passing arguments and returning values. Modular programming

Applications – Core logic for a library management system.
Learning Outcome - Ability to Understand the **different types of functions** based on parameters and return values in C Programming.

Tools/Technology to Be Used: Code:blocks

* Total Hours of Problem Definition Implementation: 3 hours
* Total Hours of Engagement = 1 hour

Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.

26.	An architect is designing a triangular garden for a residential project. To ensure the design is feasible and meets the space constraints, the architect needs to verify if the given lengths for the sides of the triangle can form a valid shape. If the sides form a valid triangle, the architect also wants to calculate its area to estimate the amount of turf or plants required. You are tasked with creating a program to assist the architect. The program should use nested functions to: • Input the side lengths: Allow the architect to input the lengths of the three sides of the proposed triangular garden. • Validate the triangle: Check if the given lengths satisfy the triangle inequality theorem (sum of any two sides must be greater than the third side). • Calculate the area using Heron's formula: If the sides form a valid triangle, compute its area using • Heron's formula: Display the results: Inform the architect whether the sides form a valid triangle and, if valid, show the area of the triangle.  © www.petervis.com	2	CO4

Test Case ID	Input (Side Lengths)	Expected Output
1.	a = 3, b = 4, c = 5	Valid Triangle: Yes Area: 6.0
2.	a = 1, b = 2, c = 3	Valid Triangle: No Message: "The given lengths do not form a valid triangle."
3.	a = -3, b = 4, c = 5	Valid Triangle: No Message: "Side lengths must be positive numbers."
4.	a = 10, b = 15, c = 25	Valid Triangle: No Message: "The given lengths do not form a valid triangle."
5.	a = 5, b = 5, c = 5	Valid Triangle: Yes Area: 10.83

Key Questions / Analysis / Interpretation to be evaluated during/after Implementation

1. Are the side lengths accepted as **floating-point** values (for precision)?
2. Is the **triangle inequality theorem** implemented correctly for all 3 combinations?
3. Is Heron's formula applied only after validation?
4. Are **functions properly nested** and reusable?
5. Is the output clear and informative for a non-technical end user (the architect)?

Supplementary Problems -

1. Add a function to classify the triangle as **Equilateral, Isosceles, or Scalene**.
2. Reject negative or zero values and display appropriate messages.
3. Extend the program to calculate for multiple triangles in a loop.
4. Add a function to display the perimeter of the triangle.

Key Skills to be addressed – Add a function to display the perimeter of the triangle.

	Applications – Landscape and garden planning **Learning Outcome -** Ability to use **functions to modularize** real-world problem in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
27.	A financial advisor is creating a tool to help clients plan their savings over a period of months. The advisor notices a common savings pattern where the amount saved in a given month is often influenced by the sum of the savings from the two previous months. This growth pattern is similar to the Fibonacci series. The advisor wants a program that: 1. Takes the number of months (n) as input from the user. 2. Generates a series showing the savings amount for each month based on this pattern. 3. Displays the series to help clients visualize how their savings might grow over time. For simplicity, assume the savings for the first two months are fixed at ₹1 each. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the input value (n) validated correctly (e.g., $n \geq 1$)? 2. Is the **Fibonacci logic** implemented accurately using loops or recursion? 3. Are the **first two months** (base cases) hardcoded to ₹1 as required? 4. Is the series displayed clearly with **month-wise labeling**? **Supplementary Problems -** 1. Let users define custom values for Month 1 and Month 2 instead of ₹1. 2. Add the total amount saved over n months and display it at the end. 3. Implement the Fibonacci logic using recursion and compare with the iterative version. 4. After calculating base savings, apply a fixed interest rate to forecast actual value. **Key Skills to be addressed –** Recursion. **Applications –** Financial planning tools (savings, investments) **Learning Outcome -** Ability to use **functions to modularize** real-world problem in C.	2	CO4
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
	Structure and Union		
28.	A librarian at CHARUSAT wants a simple digital system to manage the details of individual books in the library. The librarian frequently needs to check and update information such as the book's title, author, price, and whether it is currently issued or available.	2	CO3

	To make this process efficient, you are tasked with creating a program that uses a union to represent the details of a book. The union will optimize memory usage by storing only the necessary details of a single book at a time. The program should: • Allow the librarian to enter details for a book, including its accession number, title, author name, price, and whether the book is currently issued (use a flag: 1 for issued, 0 for available). • Display the entered book details in a readable format for verification. This program will help the librarian quickly log and review book data, ensuring smooth library operations while conserving memory. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the union correctly declared with all relevant fields? 2. Is **only one book's data** stored and displayed at a time (as intended with union)? 3. Is the **status flag** clearly handled (1 = issued, 0 = available)? 4. Is input/output logic structured, user-friendly, and readable? **Supplementary Problems -** 1. Extend the union to include book category or publisher. 2. For nesting: use a struct for title and author combined inside a union. 3. Modify the program using struct to handle an array of books and compare memory usage. 4. Add feature to update issue status based on user input (e.g., return the book = mark available). 5. Show "AVAILABLE" or "ISSUED" instead of just 1/0 using conditional display. **Key Skills to be addressed –** Union usage, formatted I/O, flag handling. **Applications –** Library/bookstore software **Learning Outcome -** Ability to **differentiate in memory handling** between union and struct in C.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
29.	At CHARUSAT, the university is organizing an inter-college sports competition and needs an efficient system to manage its sports teams. Each team represents a specific sport (e.g., basketball, football) and is led by a dedicated coach. The sports coordinator wants a program that will help maintain team and coach information in an organized manner. The system should: • Store Details: Keep a record of multiple sports teams, including the team's name, sport type, and coach information. • Manage Coach Data: Maintain detailed information about each coach, such as their name, age, and years of experience. • Allow Operations: Provide options to add new teams, search for a specific team, and display all stored teams and coach details. To achieve this, use a nested structure in C programming, where the team structure includes a sub-structure for coach information. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation**	2	CO3

	1. Has the program correctly defined a **nested structure**, where Coach is a sub-structure within Team? 2. Are **data types appropriately used** (e.g., int for age/experience, char[] for names)? 3. Can the program **store and manage multiple teams** using an array or dynamic memory? 4. Does it **prevent duplicate team entries**? 5. Are the following operations correctly implemented? a. Add a new team with complete coach details b. Search for a team by name or sport c. Display all teams and associated coach data **Supplementary Problems -** 1. Edit Team or Coach Information. 2. Delete Team Record. 3. Sort teams alphabetically by name or by coach's years of experience. 4. Save/load team and coach records to/from a file. **Key Skills to be addressed –** Union usage, formatted I/O, flag handling. **Applications –** University-level **sports team management** **Learning Outcome -** Ability to **Understand and implement nested structures in C**.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hours		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
Pointers and Dynamic Memory Allocation			
30.	A grocery store manager wants to analyze pricing trends in the inventory. You are tasked with developing a C program to assist with this task using pointers. The program should allow the manager to: 1. Input the number of items in the inventory. 2. Provide the prices of these items in an unordered manner. 3. Sort the prices in ascending order using pointer arithmetic. 4. Display the sorted prices for review. Test the created system by applying the following test cases:	4	CO5

Test Case ID	Input	Expected Output
1.	Number of items: 5 Prices: 50.5, 20.1, 35.0, 40.2, 10.0	Sorted Prices: 10.0, 20.1, 35.0, 40.2, 50.5
2.	Number of items: 6 Prices: 0.0, -10.5, 5.0, -3.2, 12.3, 8.0	Sorted Prices: -10.5, -3.2, 0.0, 5.0, 8.0, 12.3
3.	Number of items: 0	Error: "No items to sort."
4.	Number of items: 5 Prices: 9999.99, 0.01, 5000.00, 7500.50, 2500.25	Sorted Prices: 0.01, 2500.25, 5000.00, 7500.50, 9999.99
5.	Number of items: 4 Prices: 1e6, 1e5, 1e3, 1e2	Sorted Prices: 100.0, 1000.0, 100000.0, 1000000.0
6.	Number of items: 5 Prices: 50.5, 20.1, 35.0, "abc", 10.0	Error: "Invalid input for price. Please enter numeric values only."

Key Questions / Analysis / Interpretation to be evaluated during/after Implementation

1. Does the program validate that the number of items is **greater than 0**? Are **negative prices** caught and flagged as invalid?
2. Which sorting algorithm is used (Bubble, Selection, Insertion, etc.)? Is the sorting logic implemented **efficiently and correctly**?

Supplementary Problems -

1. Allow the user to sort prices in **either ascending or descending order** based on input.
2. After sorting, calculate and display **minimum**, **maximum**, and **average** price.
3. Account for **duplicate prices** and ensure sorting still works correctly.
4. Allow prices to be **read from and written to a file** for persistent storage.
5. Store and sort items **with names** (e.g., apple ₹100, banana ₹150).

Key Skills to be addressed – Array traversal and indexing, Sorting algorithms (Bubble Sort / Selection Sort)

Applications – Inventory management systems

Learning Outcome - Ability to handle and validate **real-world data input** in C. Also able to apply sorting logic to **organize and analyze numeric data** in C.

Tools/Technology to Be Used: Code:blocks

* Total Hours of Problem Definition Implementation: 1.5 hours
* Total Hours of Engagement = 2 hour

Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.

31. A content manager at a digital publishing platform is tasked with creating and editing article summaries. Initially, they allocate memory to store a short summary of the article. However, as they expand the article description, the system must dynamically adjust the memory allocation to accommodate the longer text. — 2 — CO5

1. Use calloc() to allocate memory for an initial short string (e.g., a brief article summary).
2. Allow the user to input and store the initial string.
3. Use realloc() to dynamically expand the memory allocation when the user wants to modify the string and store a larger summary.

	4. Display the updated string after reallocation. This program ensures efficient memory usage while allowing flexibility for dynamic text editing. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Does the student understand the benefit of initializing memory to zero? 2. Is the size passed to calloc() appropriate for storing the initial summary? 3. Are input functions like fgets() or scanf() used safely? 4. Are the memory reallocation steps safe (e.g., checking if realloc() returned NULL)? 5. Is the updated string displayed correctly after reallocation and modification? 6. Are memory leaks prevented by freeing memory at the end of the program? **Supplementary Problems -** 1. Allow users to **append additional content** instead of replacing the string entirely. 2. After expansion, display the number of words or characters in the summary. 3. Manage **multiple article summaries** with dynamic memory for each. 4. Write the updated summary to a **file** for permanent storage using fprintf(). **Key Skills to be addressed –** Dynamic memory allocation (calloc(), realloc()). **Applications –** Dynamic content handling in **CMS platform** **Learning Outcome -** Ability to **Understand and apply dynamic memory allocation techniques in C**.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 1.5 hours * Total Hours of Engagement = 2 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		
32.	A. A student at CHARUSAT is working on a creative writing project and needs a tool to analyze their text by reversing every word in a document. This process can help them explore unique word patterns and enhance their understanding of text structure. You are tasked with creating a program that: • Reads the content of a file named Demo.txt, which contains text input from the student. • Reverses each word in the file while maintaining the original sequence of the words. • Displays the reversed words to the user for review. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Does the program handle the case where the file does not exist? 2. Which method (fscanf, fgets, etc.) is used and why? 3. Are file operations properly closed using fclose()? 4. Does the program correctly detect spaces, tabs, or newlines as word boundaries? 5. Is each word reversed correctly without altering the word order in the sentence? 6. Is the program safe against buffer overflow when reading long lines or words? **Supplementary Problems -**	4	

	1. Option to reverse the **whole sentence** while keeping the word order intact. 2. Save the reversed content to Output.txt for later reference. 3. Show how many words were reversed in total. **Key Skills to be addressed – File handling** (fopen, fscanf, fgets, fclose) **Applications – Creative writing tools** and content transformation utilities **Learning Outcome -** Ability to u**nderstand how to perform file I/O operations in C**. B. The examination department at CHARUSAT wants a simple system to record and retrieve student marks. The system should allow faculty members to store marks in a file and retrieve them as needed for evaluation or reporting purposes. You are tasked with creating a program that: • Allows faculty to input marks for multiple students and stores them in a file using the putw() function. • Reads the stored marks from the file using the getw() function to display them for review. • Uses fopen() and fclose() for file management. This program ensures a straightforward and efficient way to manage student marks digitally, minimizing paperwork and simplifying the evaluation process. **Key Questions / Analysis / Interpretation to be evaluated during/after Implementation** 1. Is the file properly opened using fopen() in binary/write mode for putw() and read mode for getw()? 2. Is fopen() checked for failure (i.e., NULL pointer)? 3. Does the program allow input for multiple students' marks using a loop? 4. Are the marks correctly written to the file using putw()? 5. Is feof() or a similar check used properly to detect end-of-file? 6. Are the stored and retrieved marks displayed clearly and accurately? **Supplementary Problems -** 1. Store Roll Number Alongside Marks 2. Calculate Average Marks 3. Search for a Specific Student's Mark 4. Error Log File **Key Skills to be addressed – File handling** (fopen, fscanf, fgets, fclose) **Applications – Digitized student result management systems** **Learning Outcome -** Ability to u**nderstand how to perform file I/O operations in C**.		
	Tools/Technology to Be Used: Code:blocks		
	* Total Hours of Problem Definition Implementation: 3 hours * Total Hours of Engagement = 1 hour		
	Post Laboratory Work Description: Prepare a journal which contains the code and snapshot of the practical performed.		