**Practice Set**

**Chapter 1: Variables, Data types, and Input/Output**

1. Write a program to calculate the area of a rectangle. Ask the user to enter the length and width, then output the calculated area.
2. Create a program that converts temperature from Fahrenheit to Celsius. Prompt the user to enter the temperature in Fahrenheit and output the equivalent temperature in Celsius.
3. Develop a program to input a student's name and their marks in three subjects, then calculate and display their total marks and average.
4. Write a program that prompts the user to enter their age and displays it on the screen.
5. Create a program that asks the user to input their name and then prints a greeting message including their name.

**Chapter 2: Instructions & Operators**

1. Write a program that calculates the sum of two numbers entered by the user using the addition operator.
2. Develop a program to find the maximum of two numbers entered by the user using the ternary operator.
3. Create a program that calculates the area of a circle using the radius entered by the user and the value of pi (3.14).
4. Write a program to calculate the factorial of a number entered by the user using a while loop.
5. Develop a program to determine if a number entered by the user is even or odd using the modulus operator.

**Chapter 3: Conditional Statements**

1. Write a program that checks whether a number entered by the user is positive, negative, or zero using if-else statements.
2. Develop a program to find the largest among three numbers entered by the user using nested if-else statements.
3. Create a program that prompts the user to enter a character and then determines whether it is a vowel or a consonant.
4. Write a program to check if a year entered by the user is a leap year or not using if-else statements.
5. Develop a program that calculates the grade of a student based on their marks using the switch statement.

**Chapter 4: Loop Control Statements**

1. Write a program to print the multiplication table of a number entered by the user using a for loop.
2. Develop a program that finds the sum of natural numbers up to a given number using a while loop.
3. Create a program that prompts the user to enter a number and then checks if it is a prime number using a for loop.
4. Write a program to print the Fibonacci series up to a given number using a do-while loop.
5. Develop a program that counts the number of digits in a given number entered by the user using a while loop.

**Chapter 5: Functions & Recursion**

1. Write a function to calculate the power of a number raised to an exponent.
2. Develop a recursive function to find the factorial of a non-negative integer.
3. Create a program that uses a function to check if a given number is a palindrome or not.
4. Write a program that calculates the sum of digits of a number using a recursive function.
5. Develop a program that calculates the area of a triangle using a function to find the semi-perimeter.

**Chapter 6: Pointers**

1. Write a program that dynamically allocates memory for an integer variable and a pointer to that variable. Initialize the integer variable, assign its address to the pointer, and then print the value using both the variable and the pointer.
2. Develop a program that creates a structure representing a student with attributes such as name, age, and grade. Allocate memory for a structure variable dynamically, populate its attributes, and then print them using pointers.
3. Create a program that implements a simple linked list data structure to store integers. Define functions to insert elements at the beginning, end, and at a specific position, and to display the elements of the linked list.
4. Write a program to swap the values of two variables using pointers.
5. Develop a program that takes two integers as input and finds their sum, difference, product, and quotient using functions with pointers to return the results.

**Chapter 7: Arrays**

1. Write a program to find the largest and smallest elements in an array of integers entered by the user.
2. Develop a program that calculates the sum and average of elements in an array of integers.
3. Create a C program to search for a specific element in an array of integers entered by the user.
4. Write a program that sorts an array of integers in ascending order using the bubble sort algorithm.
5. Develop a program that finds the second largest element in an array of integers.

**Chapter 8: Strings**

1. Write a C program to calculate the length of a string entered by the user.
2. Develop a program that concatenates two strings entered by the user without using any built-in string functions.
3. Create a program to count the number of vowels and consonants in a string entered by the user.
4. Write a program that checks if a string entered by the user is a palindrome or not.
5. Develop a program that reverses a string entered by the user without using any built-in string functions.

**Chapter 9: Structures**

1. Write a C program to define a structure representing a student with attributes such as name, age, and roll number. Prompt the user to enter the details of two students and display them.
2. Develop a program that calculates the total and average marks of students for three subjects using a structure.
3. Create a program that sorts an array of structures representing students based on their roll numbers.
4. Write a program that finds the oldest and youngest students among a group of students represented by structures.
5. Develop a program that prompts the user to enter details of a book such as title, author, and price, and then displays the details.

**Chapter 10: File I/O**

1. Write a C program to read content from a file and display it on the console.
2. Develop a program to count the number of lines, words, and characters in a text file.
3. Create a program that reads numbers from a file, finds their sum, and writes the result to another file.
4. Write a program to copy the contents of one file to another file.
5. Develop a program to search for a specific word in a text file and display its occurrences along with the line numbers.

**Chapter 11: Dynamic Memory Allocation**

1. 16. Write a program to implement a dynamic stack using linked list. Allow the user to push and pop elements onto/from the stack and display the stack contents.
2. 17. Develop a program that dynamically allocates memory for a priority queue. Implement functions to enqueue elements with priorities and dequeue elements based on priority.
3. 18. Create a program that dynamically allocates memory for a hash table data structure. Implement functions to insert, search, and delete elements from the hash table.
4. 19. Write a program to implement a dynamic binary search tree (BST). Allow the user to insert, search, and delete nodes from the BST dynamically.
5. 20. Develop a program that dynamically allocates memory for a graph data structure. Implement functions to add vertices, add edges, and perform graph traversal algorithms like depth-first search (DFS) and breadth-first search (BFS).