C++ Programming Exercises - Set 5

This document provides a fifth set of problem statements for learning C++ programming. Each set is designed to challenge students with progressively complex tasks, ranging from basic operations to advanced programming concepts such as polymorphism and templates. Each category includes problems that build on previously covered principles, introducing new complexities and advanced scenarios.

Beginners - Set 5

s5Beginner1.cpp - Write a program that takes two integer inputs, performs all arithmetic operations (addition, subtraction, multiplication, division, modulus), and prints the results.

s5Beginner2.cpp - Modify the program to accept user inputs in a loop until the user types 'exit'. Each time, perform and print the results of the arithmetic operations.

s5Beginner3.cpp - Extend the program to include error handling that checks for and handles division by zero and invalid input data types.

s5Beginner4.cpp - Enhance the program by calculating the square and cube of each input number, and also determining if the input numbers are even or odd.

Intermediates - Set 5

s5Intermediate1.cpp - Create a program that reads an array of integers and finds the maximum and minimum values.

s5Intermediate2.cpp - Modify the program to allow the user to specify the size of the array and the elements, then perform the max/min evaluation.

s5Intermediate3.cpp - Extend the program to sort the array in ascending and descending order using simple sorting algorithms (e.g., bubble sort).

s5Intermediate4.cpp - Add functionality to the program to compute the mean, median, and mode of the array elements.

Advanced - Set 5

s5Advanced1.cpp - Define a class `Vehicle` with properties and methods appropriate for a vehicle like make, model, and year. Include methods to display this information.

s5Advanced2.cpp - Extend the `Vehicle` class with derived classes for `Car` and `Truck`, adding unique properties and overriding display methods.

s5Advanced3.cpp - Implement a simple dynamic memory management feature in the `Vehicle` class, ensuring proper constructor and destructor handling.

s5Advanced4.cpp - Add a static member to the `Vehicle` class to keep track of the number of vehicle objects created and destroyed throughout the program execution.

Experts - Set 5

s5Expert1.cpp - Develop a template class `Matrix` that can handle mathematical operations (addition, subtraction, multiplication) on matrices of any numeric data type.

s5Expert2.cpp - Extend the `Matrix` template class to include matrix transpose and determinant calculation features.

s5Expert3.cpp - Implement exception handling in the `Matrix` class to manage errors like invalid matrix dimensions for operations.

s5Expert4.cpp - Create a user interface in the main function that allows users to dynamically create matrices, perform operations, and display results using the `Matrix` class methods.