C++ Intelligence Test Exam

Duration: 3 Hours

Instructions

- This exam consists of 5 problems.
- The total marks for this exam are 100.
- Write your answers in the space provided.
- For coding questions, ensure your code is syntactically correct, efficient, and well-structured.
- Time allotted: 3 hours.

Problem 1: Optimizing Memory and Performance (20 Marks)

You are given a large array of integers (e.g., 1 million elements). Write a C++ program that: 1. Finds the top 10 largest unique numbers in the array. 2. Ensures the program uses minimal memory and runs in optimal time complexity. 3. Uses STL containers and algorithms effectively.

Provide the code and explain your approach to optimization.

Answer: __________

—

Problem 2: Recursive Algorithms and Templates (20 Marks)

Write a C++ template function ‘recursive_power‘ that calculates x^n (where n is a non-negative integer) using **recursion**. The function should: 1. Handle any data type (e.g., ‘int‘, ‘double‘, ‘float‘). 2. Optimize the number of recursive calls using the **divide-and-conquer** approach (e.g., $x^n = x^{n/2} \times x^{n/2}$).

Write a ‘main()‘ function to demonstrate the usage of this template function.

Answer: _________

—

Problem 3: Custom Iterator (20 Marks)

Create a custom iterator class ‘RangeIterator‘ that iterates over a range of numbers (e.g., from ‘start‘ to ‘end‘ with a given ‘step‘). The iterator should: 1. Support the ‘++‘, ‘–‘, ‘*‘, and ‘!=‘ operators. 2. Be compatible with STL algorithms like ‘std::for_each‘. Write a ‘main()‘ function to demonstrate the usage of ‘RangeIterator‘ with a lambda function to print all numbers in a range.

Answer: _________

—

Problem 4: Advanced Lambda Functions and STL (20 Marks)

Write a C++ program that uses ‘std::vector‘ to store a list of strings. Perform the following tasks: 1. Use a lambda function to sort the strings by their length in ascending order. 2. Use another lambda function to filter out all strings that contain a specific substring. 3. Use ‘std::accumulate‘ and a lambda function to concatenate all remaining strings into a single string.

Provide the complete code and explain your approach.

Answer: _________

—

Problem 5: Designing a Custom Container (20 Marks)

Design and implement a custom container class ‘CircularBuffer‘ that represents a circular buffer (fixed-size queue). The class should: 1. Support operations like ‘push‘, ‘pop‘, ‘front‘, and ‘size‘. 2. Use a dynamically allocated array internally. 3. Handle edge cases (e.g., buffer full, buffer empty). 4. Provide an iterator to traverse the buffer.

Write a ‘main()‘ function to demonstrate the usage of the ‘CircularBuffer‘ class.

Answer: __________

—