C++ Quiz: Intermediate and Advanced Level

Instructions

Answer the following 30 questions to the best of your ability. Each question is designed to test your understanding of C++ at an intermediate to advanced level. You may use the C++ Standard Library where applicable.

Quiz Questions

Q1. What is the output of the following code snippet? Explain the behavior of the ‘std::vector‘ in this case:

```
std::vector<int> v = {1, 2, 3};
for (auto it = v.begin(); it != v.end(); ++it) {
    if (*it == 2) {
        v.erase(it);
    }
}
```

Q2. How does the C++ compiler distinguish between function overloading and template specialization? Provide an example for each.

Q3. Consider the following code. Explain why it causes undefined behavior:

```
int* ptr = new int[5];
delete ptr;
```

Q4. What are the key differences between ‘std::unique_ptr‘ and ‘std::shared_ptr‘? Provide a situation where each would be appropriate.

Q5. What is the output of the following code? Explain why:

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> v(3, 10);
    v.reserve(10);
    std::cout << v.size() << " " << v.capacity() << std::endl;
    return 0;
}
```

Q6. Write a lambda function to sort a vector of ‘std::pair¡int, int¿‘ by the second element in descending order.

Q7. What is the purpose of ‘std::enable_if‘ in C++ templates? Provide an example of its usage.

Q8. Explain the difference between ‘std::move‘ and ‘std::forward‘. In what scenarios is each used?

Q9. Analyze the following code. What will be the output, and why?

```
#include <iostream>
#include <thread>

void printMessage() {
    std::cout << "Hello from thread!\n";
}

int main() {
    std::thread t(printMessage);
    t.detach();
    return 0;
}
```

Q10. How does ‘std::atomic‘ ensure thread safety in multithreaded programs? Illustrate with a code example.

Q11. What are the differences between ‘virtual‘, ‘override‘, and ‘final‘ in C++? Provide a code example demonstrating their usage.

Q12. Explain the concept of ‘perfect forwarding‘. Write a function template that demonstrates perfect forwarding.

Q13. Why are ‘std::string‘ iterators invalidated when the string is resized? Provide a code example illustrating this behavior.

Q14. What is the difference between ‘std::map‘ and ‘std::unordered_map‘ in terms of performance and usage? Provide an example.

Q15. Discuss the use and implications of ‘volatile‘ in C++.

Q16. Consider the following code snippet. What will be the output, and why?

```
#include <iostream>
class A {
public:
    A() { std::cout << "Constructor\n"; }
    ~A() { std::cout << "Destructor\n"; }
};

int main() {
```

```
    A* obj = new A[2];
    delete obj;
    return 0;
}
```

Q17. What is the output of the following code? Explain:

```
#include <iostream>
class A {
    int x;
public:
    A(int val) : x(val) {}
    int getValue() const { return x; }
};

int main() {
    const A obj(10);
    std::cout << obj.getValue() << std::endl;
    return 0;
}
```

Q18. Implement a singleton class in C++ using modern C++ techniques.

Q19. Explain the difference between ‘new‘, ‘malloc‘, and ‘std::make_shared‘.

Q20. Write a program to demonstrate the use of ‘std::weak_ptr‘. How is it different from ‘std::shared_ptr‘?

Q21. Why is ‘std::vector¡std::atomic¡int¿¿‘ not allowed to be resized using ‘resize()‘? Provide an example and explanation.

Q22. What are the implications of using ‘inline‘ functions in C++? Provide an example where ‘inline‘ may not behave as expected.

Q23. Write a C++ program to demonstrate the use of ‘std::condition_variable‘.

Q24. Explain the differences between ‘emplace_back‘ and ‘push_back‘ in a ‘std::vector‘. Provide examples.

Q25. Write a program to count the number of words in a given string using ‘std::istringstream‘.

Q26. What is the difference between ‘std::optional‘ and ‘std::variant‘? Provide examples demonstrating their usage.

Q27. Implement a function template to calculate the dot product of two vectors in C++.

Q28. Write a program to find the longest common subsequence between two strings using dynamic programming.

Q29. What is the output of the following code, and why? Explain:

```
#include <iostream>
#include <string>

int main() {
    std::string s = "Hello";
    s += " World";
    std::cout << s << std::endl;
    return 0;
}
```