CSE232 US22 Final Exam (Practice)

This is a practice exam for the CSE232 final exam. Note: these questions are similar in nature but not identical to the actual final exam.

1. **Which of the following are true about a lambda function?**
   (a) It is a nameless function
   (b) They are often used in conjunction with STL functions
   (c) They have a capture list
   (d) All the above
   (e) None of the above

2. **Which of the following are true about a function which is templated?**
   (a) It is itself not a function, but a way to create a function
   (b) It contains the keyword `template`
   (c) It makes use of a template parameter to represent a calling type
   (d) All the above
   (e) None of the above

3. **Which of the following are true about a function parameter which is both const and a reference?**
   (a) A copy is made of that parameter when the function is invoked
   (b) You cannot change the parameter inside the function
   (c) It only works with pointer types
   (d) All the above
   (e) None of the above
4. **Which of the following describes the term “memory leak?”**
   (a) An undeclared array
   (b) A pointer that points to the value `nullptr`
   (c) Memory that was allocated by `new` but not deleted with `delete`
   (d) Memory that is untyped
   (e) None of the above

5. **What is the return type of a class constructor?**
   (a) An instance of the class
   (b) No return on a constructor
   (c) An int (the size of the class in bytes)
   (d) A string (the name of the class)
   (e) None of the above

6. **Which of the following is true about the special variable “this?”**
   (a) It is a reference value
   (b) You are allowed to change what it points to within a method
   (c) It is point to the calling instance of a method
   (d) All of the above
   (e) None of the above

7. **What is the “Rule of Three” for C++ classes?**
   (a) If you provide a copy constructor, destructor, or operator=, you should provide all of them.
   (b) You must provide three constructors in a dynamic memory class
   (c) There should be no more than three private variables in a class
   (d) You need a destructor if the size of a private data member dynamic array is bigger than 3.
   (e) None of the above
Note: Refer to the following code snippet for questions 8 - 9.

```cpp
int fn1(const std::vector<std::vector<int>>& vec) {
    int result = 0;  // Line 1
    for (auto row : vec) {  // Line 2
        for (auto col : row) { // Line 3
            if (col % 2)          // Line 4
                result += col;   // Line 5
        }
    }
    return result;
}
```

8. What type is the variable `row` in `fn1` on Line 2?

9. If `vec = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}`, what is the result of `fn1(vec)`?

Note: Refer to the following code snippet for questions 10 - 11.

```cpp
std::vector<int> fn2(const std::vector<int>& vec1,
                     const std::vector<int>& vec2) {
    std::vector<int> result;  // Line 1
    for (auto e : vec1) {      // Line 2
        auto i = std::find(vec2.begin(), vec2.end(), e); // Line 3
        if (i != vec2.end()) {
            result.push_back(e);
        }
    }
    return result;
}
```

10. If `vec1 = {0, 1, 2, 3}` and `vec2 = {2, 3, 4, 5}`, what is the result of calling `fn2(vec1, vec2)`?

11. If `vec1 = {4, 5, 3, 8}` and `vec2 = {8, 3, 2, 4}`, what is the result of calling `fn2(vec1, vec2)`?
Note: Refer to the following code snippet for questions 12 - 15.

```cpp
std::map<int, std::string> fn3(const std::vector<int>& vec) {
    std::map<int, std::string> result;                         // Line 1
    std::transform(vec.begin(), vec.end(),                  // Line 2
                    std::inserter(result, result.end()),   // Line 3
                    [](auto x) {                           // Line 4
                        std::string v = std::to_string(x);  // Line 5
                        std::reverse(v.begin(), v.end());  // Line 6
                        return std::pair<int, std::string>(x, v); // Line 7
                    });
    return result;
}
```

12. **What is the type of the variable x on Line 4?**

13. **If we set result = fn3({1, 101, 201, 331, 5659, 23079}), what is the value of result[101]?**

14. **If we set result = fn3({1, 101, 201, 331, 5659, 23079}), what is the value of result[5659]?**

15. **If we set result = fn3({1, 101, 201, 331, 5659, 23079}), what is the value of result[23079]?**

Note: Refer to the following code snippet for questions 16 - 19.

```cpp
std::vector<std::string> fn4(const std::vector<std::string>& vec, int n) {
    std::vector<std::string> result;                         // Line 1
    std::copy_if(vec.begin(), vec.end(),                    // Line 2
                  std::back_inserter(result),            // Line 3
                  [&n](auto x) {                         // Line 4
                      int m = static_cast<int>(x.size()); // Line 5
                      return (m == n);                   // Line 6
                  });
    return result;
}
```

16. **What is the type of the variable x on Line 4?**
17. If is the value returned by \( \text{fn4}(["cat", "dog", "duck", "goat", "goose"], 4) \)?

18. If is the value returned by \( \text{fn4}(["cat", "dog", "duck", "goat", "goose"], 1) \)?

19. If is the value returned by \( \text{fn4}(["cat", "dog", "duck", "goat", "goose"], 5) \)?

Note: Refer to the following code snippet for questions 20 - 23.

```cpp
std::vector<int> fn5(int n) {
    int a = 0;
    int b = 1;
    std::vector<int> result(n);
    std::generate(result.begin(), result.end(),
                   [&a, &b]() {
                       int c = b;
                       b = a + b;
                       a = c;
                       return a;
                   });
    return result;
}
```

20. What is the result of \( \text{fn5}(1) \)?

21. What is the result of \( \text{fn5}(3) \)?

22. What is the result of \( \text{fn5}(5) \)?

23. What is the result of \( \text{fn5}(7) \)?
#include <iostream>

class Point {
private:
    double x_ = -1; // Line 1
    double y_ = -1; // Line 2

public:
    Point() = default; // Line 3
    Point(double x, double y) : x_(x), y_(y) {} // Line 4
    Point(const Point& p) : x_(p.x_), y_(p.y_) {} // Line 5
    ~Point(); // Line 6
    double x() const { return x_; } // Line 7
    double y() const { return y_; } // Line 8
    void x(double x) { x_ = x; } // Line 9
    void y(double y) { y_ = y; } // Line 10
    Point& operator=(const Point& p) { // Line 11
        x_ = p.x_;
        y_ = p.y_;
        return *this;
    }

friend double norm(const Point& p1, const Point& p2); // Line 12
};

double norm(const Point& p1, const Point& p2) {
    return sqrt(pow(p1.x_ - p2.x_, 2) + pow(p1.y_ - p2.y_, 2));
}

int main() {
    Point p0 = Point(); // Line 13
    Point p1 = Point(0, 0); // Line 14
    Point p2 = Point(3, 4); // Line 15
    double result = norm(p1, p2); // Line 16

    return 0;
}

24. **What is the meaning of the default keyword on Line 3?**

25. **Which Lines implement the “Rule of Three” class member functions?**

26. **What does the const modifier on Lines 7 and 8 mean for the associated functions?**
27. What is the meaning of the friend keyword on Line 12?

28. What is the value of \( p_0.x \) and \( p_0.y \)?

29. What is the value of \( p_1.x \) and \( p_1.y \)?

30. What is the value of \( p_2.x \) and \( p_2.y \)?

31. What is the value of result?