CS205 C/CPP Design Quiz Problem

Note:

- Quiz has the following types of questions:
    - judgment questions (Give a statement, judge true or false)
    - Single choice/multiple choice questions (Give questions and options and select one or more of the correct items)
    - blank filling questions (Fill in a simple answer to the question)
- This material may be helpful for your class quizzes and final exams, so please keep it well.
- The producer of this material is Maystern.

1. The source code of program is as follows:

```
int main(int arc, char *argv[]) {
    ...
}
```

   The source code has been compiled to program ***hello*** , when you run it as follows, what ***argc*** will be ?

```
./hello I LOVE CPP
```

2. The following code is the ***prototype/declaration*** of function ***mul*** .

```
int mul(int a, int b) {
    return a * b;
}
```

3. The data type of the variable ***PI*** is ***double*** .

```
#define PI 3.14
```

4. When a function prototype is declared in the header file you create, you do NOT need to define it in a CPP file.
5. IDE (Integrated development environment) is a powerful compiler with many useful features.
6. If you get 75 for one of your projects, which situation should most likely be?

   A. Finish all tasks almost perfectly

   B. Finish all tasks well

   C. Correct

   D. Finish all tasks

7. What's the output of the source code?

```
signed char c1 = 127;
signed char c2 = 1;
int csum = c1 + c2;
cout << csum;
```

8. `sizeof()` is a function and can yield the size in bytes of a type.

```
int int_num = 10;
size_t len = sizeof(int_num);
```

9. `size_t` is an unsigned integer type.
10. What's the value of the variable ***num1, num2, num3*** ?

```
int num1 = 23 / 4 * 4;
int num2 = 23 / 4 * 4.;
int num3 = 23 / 4. * 4;
```

11. In the following code, since the variable num is not initialized explicitly, it will be initialized to 0 automatically.

```
int num;
cout << num << endl;
```

12. auto is a placeholder type specifier in C++11. What is the value of the variable ***val*** in the following code?

```
auto val = 2 / 3;
val = 3.14 * 2;
```

13. What's the output of the following source code ?

```
int x = 100;
int y = 30;
x += (y -= 10);
cout << x;
```

14. What's the output of the following source code ?

```
int x = 100;
int y = x++;
cout << y;
```

15. What's the output of the following source code ?

```
int x = 100;
int y = (x = 200);
cout << y;
```

16. What's the output of the following source code ?

```
int a = 2, b = 0;
if (a || b++) {
    cout << b;
}
```

17. What's the output of the following source code ?

```
int i = 0;
for (i = 1; i < 100; i += 2) {
    // something
}
cout << i
```

18. What's the output of the source code ?

```
int x = 5;
do {
    x = 0;
} while (x);
cout << x;
```

19. The following source code (empty in the parentheses) can be compiled successfully.

```
for () {
    // some lines here
}
```

20. What is the output of the following code ?

A. Key 4

B. Key 5

C. Undefined key.

```
int num = 5;
switch (num) {
    case '4':
        cout << "Key 4" << endl;
    case '5':
        cout << "Key 5" << endl;
    default:
        cout << "Undefined key."
}
```

21. How many lines will be printed ?

A. Compilation error

B. 10

C. 11

D. None of the above

```
for (size_t i = 10; i ⩾ 0; i--)
    cout << "Line " << i << endl;
```

22. What's the output of the source code?

A. 2

B. 4

C. 6

D. 8

E. The source code cannot be compiled without error.

F. Unpredictable result.

```
#include <iostream>
using namespace std;
int main() {
    int idx = 0;
    int numbers[4] = {2, 4, 6, 8};
    idx = -1;
    cout << numbers[idx];
}
```

23. What's the output of the source code?

A. 2

B. 4

C. 6

D. 8

E. The source code cannot be compiled without error.

F. Unpredictable result.

```
#include <iostream>
using namespace std;
int main() {
    int idx = 0;
    int numbers[4] = {2, 4, 6, 8};
    idx = 4;
    cout << numbers[idx];
}
```

24. What's the output if you compile the following source code with C++11 standard?

```
char str[16] = {"C++"};
cout << strlen(str) << endl;
```

25. What's the output if you compile the following source code with C++11 standard?

```
char str {"C++"};
cout << str[1] << endl;
```

26. What's the output?

A. C++

B. No output.

C. Compilation error.

D. Runtime error.

```
char str1[16] = {"C++"};
char str2[16];
str2 = str1;
cout << str2;
```

27. What's the output?

A. C++

B. No output.

C. Compilation error.

D. Runtime error.

```
string str1 = {"C++"};
string str2;
str2 = str1;
cout << str2;
```

28. What's the output?

```
#include <iostream>
union twonumbers {
    int n[2];
    double d;
};
using namespace std;
int main () {
    twonumbers tn;
    tn.n[0] = 0;
    tn.n[1] = 0;
    cout << tn.d;
    return 0;
}
```

29. The output of the following code is:

```
struct Person {
    bool male;
    int id;
    char label;
}
cout << sizeof(struct Person);
```

30. What's the output of the following code?

```
int *numbers = new int[8];
char *pc = (char *) numbers;
*numbers = 0x0A0B0C0D;
cout << (int) pc[3];
```

31. The following source code is correct.

```
int *ptr;
*ptr = 3;
```

32. What's the output of the following code on your PC (64 bit OS and CPU)?

```
int numbers[8];
cout << sizeof(numbers) << endl;
```

33. What's the output of the following code on your PC (64 bit OS and CPU)?

```
int *numbers = new int[8];
cout << sizeof(numbers) << endl;
```

34. The following code can be compiled successfuly.

```
double value = 0.0;
const double *p = &value;
value = 2.0;
```

35. The following code can be compiled successfuly.

```
double value = 0.0;
double * const p = &value;
p[0] = 2.0;
```

36. The following source code is correct and cannot cause bugs.

```
int *pint = (int *) malloc(8 * sizeof(int));
char * pc = (char *) pint;
pc[8] = 'a';
*(pc + 8) = 'b';
```

37. What's the output of the following code?

```
# include <iostream>
using namespace std;
void foo(float * p) {
    p[0] = 1.0f;
}
int main() {
    float values[4] = {3.0f, 4.0f, 5.0f, 6.0f};
    foo(values + 2);
    cout << *values << " ";
    cout << *values + 2 << " ";
    cout << *(values + 2) << " ";
    cout << values[2] << endl;
    return 0;
}
```

38. What's the output of the following code?

```
# include <iostream>
using namespace std;
struct people {
    string name;
    int age;
};
void init(people p) {
    p.name = "No name";
    p.age = 0;
}
int main() {
    people p;
    p.age = -1;
```

```
    init(p);
    cout << p.age << endl;
    return 0;
}
```

39. What's the output of the following code?

A. 0

B. -1

C. < Compile Error >

D. Random Value

```
# include <iostream>
using namespace std;
struct people {
    string name;
    int age;
};
void init(people * p) {
    p.name = "No name";
    p.age = 0;
}
int main() {
    people p;
    p.age = -1;
    init(p);
    cout << p.age << endl;
    return 0;
}
```

40. What's the output of the following code?

```
# include <iostream>
using namespace std;
struct people {
    string name;
    int age;
};
void init(people * p) {
    p -> name = "No name";
    p -> age = 0;
}
int main() {
    people p;
    p.age = -1;
    init(&p);
    cout << p.age << endl;
    return 0;
}
```

41. What's the output of the following source code?

```
# include <iostream>
using namespace std;
```

```
struct people {
    string name;
    int age;
};
void init(people & p) {
    p.name = "No name";
    p.age = 0;
}
int main() {
    people p;
    p.age = -1;
    init(p);
    cout << p.age << endl;
    return 0;
}
```

42. What's the output of the following code?

```
# include <iostream>
using namespace std;
float area(float & x) {
    return x * x;
}
int main() {
    float value = 3.0f;
    cout << area(value);
}
```

43. What's the output?

```
# include <iostream>
using namespace std;
int area(int &x);
int main() {
    int n = 10;
    area(n);
    cout << n << endl;
}
int area(int &x) {
    return x *= x;
}
```

44. What's the output of the following code?

```cpp
# include <iostream>
using namespace std;
float area(float & x) {
    x = x * x;
    return x;
}
int main() {
    float value = 3.0f;
    cout << area(value) << endl;
    cout << value << endl;
    return 0;
}
```

45. What's the output of the following code?

A. 3

B. 9

C. < Compile Error >

D. < None of the above >

```cpp
# include <iostream>
using namespace std;
float area(float * x) {
    return *x * *x;
}
int main() {
    float value = 3.0f;
    cout << area(value);
    return 0;
}
```

46. What's the output of function ***sum()*** ?

A. 4 or 8

B. 3

C. 12

D. 32

E. < Compile Error >

```cpp
# include <iostream>
using namespace std;
int main() {
    int cookies[8] = {1, 2, 4, 8, 16, 32, 64, 128};
    int n = sum_arr(cookies, 3);
}
int sum_arr(int arr[], int n) {
    cout << sizeof arr;
    return 0;
}
```

47. What's the output of function ***sum()*** ?

A. 4 or 8

B. 3

C. 12

D. 32

E. < Compile Error >

```
# include <iostream>
using namespace std;
int sum_arr(int arr[], int n) {
    cout << sizeof arr;
    return 0;
}
int main() {
    int cookies[8] = {1, 2, 4, 8, 16, 32, 64, 128};
    int n = sum_arr(cookies, 3);
}
```

48. There is a function template. The specialization is correctly implemented in the following code.

```
template <typename T>
T sum(T x, T y) {
    return x + y;
}
struct Point {
    int x;
    int y;
};
template Point sum<Point>(Point pt1, Point pt2) {
    Point pt;
    pt.x = pt1.x + pt2.x;
    pt.y = pt1.y + pt2.y;
    return pt;
}
```

49. The following declaration correctly defines some default arguments.

```
int harpo(int n = 3, int m, int k = 3);
```

50. The functions and a function pointer are declared as follows. Which answers are correct?

```
float norm_l1(float x, float y); //declaration
float norm_l2(float x, float y); //declaration
float (*norm_ptr)(float x, float y); //norm_ptr is a function pointer
```

A. `norm_ptr = &norm_l1;`

B. `norm_ptr = norm_l2;`

C. `norm_ptr = norm_l2; norm_ptr(3.0f, 4.0f);`

D. `norm_ptr = &norm_l1; (*norm_ptr)(3.0f, 4.0f);`

E. `norm_ptr = norm_l1; (*norm_ptr)(3.0f, 4.0f);`

51. The following code correctly defines a function template:

```
template <typename T>
void swap(T &a, T &b) {
    T temp = a;
    a = b;
    b = temp;
}
```

52. Function overloading is that multiple functions share the same function name but different signatures as the two functions below:

```
float foo(float arg);
int foo(double arg);
```

53. The ***this*** pointer points to the object and can be used to invoke a member as in the following code.

```
class Person {
    int num;
    public:
        static int foo() {
            return this→num;
        }
        // other members
};
```

54. What's the output of the following code?

A. No name

B. < No output >

C. < Compilation error >

D. < Runtime error >

```
class Person {
    string name;
};
int main() {
    Person p;
    p.name = "No name";
    cout << p.name;
    return 0;
}
```

55. What's the output of the following source code?

```
class Hello {
    static int value;
    int num;
    public:
        int sum(int i, int j) {
            return i + j;
        }
};
int main() {
    cout << sizeof(Hello) << endl;
    return 0;
}
```

56. What's the output of the source code?

```
class Hello {
    static int value;
    int num;
    public:
        int sum(int i, int j) {
            return i + j;
        }
        void setValue(int v) {
            value = v;
        }
        int getValue() {
            return value;
        }
};
int main() {
    Hello h1, h2;
    h1.setValue(5);
    cout << h2.getValue() << endl;
    return 0;
}
```

A. 5

B. A random integer

C. Compilation error

D. Link error

57. A class is declared as follows. Please select correct answers for creating a variable.

```
class Stock{
    public:
        Stock();
        Stock(const std::string & co, long n = 0, double pr = 0.0);
        ~Stock();
        //other members
};
```

A. Stock st1;

B. `Stock st2("MSFT", 3, 2.0f);`

C. `Stock st3 = Stock("MSFT", 3, 2.0f);`

58. Class ***Stonewt*** is declared as follows.

```
class Stonewt {
    // some members
    public:
        Stonewt(double lbs);
        Stonewt(int stn, double lbs);
        Stonewt();
        ~Stonewt();
        operator int() const;
        operator double() const;
}
```

Which function will be invoked by the following line of code ?

```
Stonewt wt = 120;
```

59. Which function will be invoked by the following line of code ?

```
wt = 120.0; //wt is an object of type Stonewt
```

60. Which function will be invoked by the following line of code ?

```
double f = wt; //wt is an object of type Stonewt
```

【分析】`wt` 是 Stonewt 类的一个对象，使用 `operator double() const;` 进行隐式类型转换。

【答案】`operator double() const;`

61. Which function will be invoked by the following line of code ?

```
Stonewt wt(120);
```

62. A conversion function is defined outside of the declaration of class ***Stonewt*** as follow.

```
Stonewt::operator double() const {
    return pounds;
}
```

63. Assignment operator '=' can be overloaded by a non-member function.

64. We can change operators' precedence by overloading.

65. If the friend function is defined as in the following source code, it is a member function in the class.

```
class Time {
    private:
        int hours;
        int minutes;
    public:
        Time();
        Time(int h, int m = 0);
```

```
        void AddMin(int m);
        void AddHr(int h);
        void Reset(int h = 0, int m = 0);
        Time operator + (const Time & t) const;
        Time operator - (const Time & t) const;
        Time operator * (double n) const;
        void show() const;
        friend Time operator * (double mult, Time in);
};
```

66. If we define a member function as follows for class Time

```
Time Time::operator*(double mult) const
```

then we can calculate as follows

```
a = 3.3 * b;//a and b are objects of type Time
```

67. operator+() overloads the + operator, and it can only be used for mathematical addition.
68. You can define two constructors as follows for class Person.

```
Person(){...}
Person(int m = 0) {...}
```

69. Please read the following code and choose correct answers:

```
class Person {
    char *name;
    public:
        Person() {
            name = new char [128];
        }
        ~Person() {
            delete name;
        }
};
```

A. The code can be compiled without error.

B. Runtime error.

C. It can cause memory double free problem.

D. It can cause memory leak.

70. For class Person, which of the constructors is a copy constructor?

A. `Person::Person();`

B. `Person::Person(const Person & p);`

C. `Person::Person(int m);`

D. `Person::Person(int m, int n);`

71. For class Person, which of the constructors is its default constructor?

A. `Person::Person();`

B. `Person::Person(const Person & p);`

C. `Person::Person(int m);`

D. `Person::Person(int m, int n);`

72. If assignment operator is not defined in class Person, the following code will invoke default assignment operator.

```
p1 = p2 = p3; //p1, p2 and p3 are objects of type Person
```

73. If you do not define a default constructor for a class explicitly, then no default constructor for that class.

74. What is the output?

```
class Animal {
    private:
        int weight;
    public:
        Animal(int w = 0) {
            weight = w;
        }
        void print() {
            cout << weight << endl;
        }
};
class Dog: public Animal {
    public:
        Dog(int w = 0): Animal(w) {}
        void print() {
            cout << "Dog ";
            Animal::print();
        }
        void speak() {
            cout << "wangwang" << endl;
        }
};
int main() {
    Dog dog(5);
    Animal * p = & dog;
    p -> print();
    return 0;
}
```

75. What is the output?

A. wangwang

B. < Compilation Error >

C. None of above

```
class Animal {
    private:
        int weight;
    public:
        Animal(int w = 0) {
            weight = w;
        }
        void print() {
            cout << weight << endl;
```

```
        }
};
class Dog: public Animal {
    public:
        Dog(int w = 0): Animal(w) {}
        void print() {
            cout << "Dog ";
            Animal::print();
        }
        virtual void speak() {
            cout << "wangwang" << endl;
        }
};
int main() {
    Dog dog(5);
    Animal * p = & dog;
    p -> speak();
    return 0;
}
```

76. What is the output?

```
class Animal {
    private:
        int weight;
    public:
        Animal(int w = 0) {
            weight = w;
        }
        virtual void print() {
            cout << weight << endl;
        }
};
class Dog: public Animal {
    public:
        Dog(int w = 0): Animal(w) {}
        void print() {
            cout << "Dog ";
            Animal::print();
        }
        void speak() {
            cout << "wangwang" << endl;
        }
};
int main() {
    Dog dog(5);
    Animal * p = & dog;
    p -> print();
    return 0;
}
```

77. Please choose the right answer(s) for declaring a class template

A. template <class Type> class ClassName{...}

B. template <typename Type> class ClassName{...}

78. Matx and Matx12f are declared in the following figure. Please choose the correct statement(s).

A. Matx is a class tempate.

B. Matx is a tempate class.

C. Matx12f is a class template.

D. Matx12f is a template class.

```
template <typename _Tp, int m, int n> class Matx {
    public:
        enum {
            rows = w,
            cols = n,
            channels = rows * cols,
#ifdef OPENCV_TRAITS_ENABLE_DEPRECATED
            depth = traits::Type<_Tp>:: value,
            type = CV_MAKETYPE(depth, channels),
#endif
            shortdim = (m < n ? m : n)
        };
};
typedef Matx<float, 1, 2> Matx12f;
typedef Matx<double, 1, 2> Matx12d;
typedef Matx<float, 1, 3> Matx13f;
typedef Matx<double, 1, 3> Matx13d;
typedef Matx<float, 1, 4> Matx14f;
typedef Matx<double, 1, 4> Matx14d;
typedef Matx<float, 1, 6> Matx16f;
typedef Matx<double, 1, 6> Matx16d;
```

79. What's the output of the following code?

```
#include <iostream>
using namespace std;
template <typename Type>
class Stack {
    private:
        Type items[16];
    public:
        size_t size() {
            return sizeof(items);
        }
};
int main() {
    Stack<float> st;
    cout << st.size() << endl;
    return 0;
}
```

80. When an exception is thrown, the program must be terminated.

81. A try block can be followed by multiple catch blocks.

82. The following source code cannot be compiled successfully.

```
double gmean(double a, double b) {
    if (a < 0 || b < 0)
        throw string("bad arguments");
    return std::sqrt(a * b);
}
int main() {
    try {
        gmean(3, -3);
    }
}
```

83. The following code cannot be compiled successfully since 'try' is commented.

```
double gmean(double a, double b) {
    if (a < 0 || b < 0)
        throw string("bad arguments");
    return std::sqrt(a * b);
}
int main() {
    gmean(3, -3);
}
```