



# Software Engineering

Home ► My courses ► Previous Years ► 2020-21 ► Spring Semester (2020-21) ►  
Software Engineering ► Theory Quiz -2 ► Theory Quiz - 2

**Started on** Wednesday, 24 February 2021, 2:15 PM

**State** Finished

**Completed on** Wednesday, 24 February 2021, 3:05 PM

**Time taken** 50 mins 46 secs

**Grade** 14.50 out of 15.00 (97%)

## Question 1

Correct

Mark 1.00 out of  
1.00

Flag question

Match the *purpose* with the *cast operator* between the lists.

### Option# Purpose

- (a) safe down cast on a polymorphic hierarchy
- (b) safe up cast on inheritance hierarchy
- (c) cast a pointer to an **int** and vice-versa
- (d) change cv-qualifier of pointers and references

and the outcome from the program as filled up.

### Option# Cast Operator

- (1) const\_cast
- (2) static\_cast
- (3) dynamic\_cast
- (4) reinterpret\_cast

change cv-qualifier of pointers and references  ✓

safe down cast on a polymorphic hierarchy  ✓

safe up cast on inheritance hierarchy  ✓

cast a pointer to an **int** and vice-versa  ✓

Your answer is correct.


The correct answer is: change cv-qualifier of pointers and references

- const\_cast, safe down cast on a polymorphic hierarchy
- dynamic\_cast, safe up cast on inheritance hierarchy
- static\_cast, cast a pointer to an **int** and vice-versa
- reinterpret\_cast

## Question 2

Partially correct

Mark 0.50 out of 1.00

 Flag question

Consider the program below where LINE 1, LINE 2, and LINE 3 are commented. As we un-comment these lines we see different outcomes.

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

class MyException : public exception {
    string msg;
    //MyException(const MyException& e);    // LINE 1
public:
    //MyException(const MyException& e): msg("Copied " + e.ms
g) { }    // LINE 2
    //MyException(const MyException& e);    // LINE 3
    MyException(): msg("Exception") { }
    const char* what()
    { return msg.c_str(); }
};

void g() {
    throw MyException();
}

void f() {
    try { g(); }
    catch (MyException& ex) {
        throw;
    }
}

int main() {
    try { f(); }
    catch (MyException ex) {
        cout << string(ex.what()) << endl;
    }
    return 0;
}
```

Match the un-commenting with the outcome:

Line Un-  
commented

(a) None

(b) LINE 1

(c) LINE 2

- (a) None  
(d) LINE 3

Program Outcome

- (1) Prints Exception  
(2) Unresolved external  
(3) Cannot access private member  
(4) Prints Copied Exception

LINE 2	<input type="text" value="Prints Exception"/>	✗
LINE 1	<input type="text" value="Cannot access private member"/>	✓
LINE 3	<input type="text" value="Prints Exception"/>	✗
None	<input type="text" value="Prints Exception"/>	✓

Your answer is partially correct.

You have correctly selected 2.

The correct answer is: LINE 2

- Prints Copied Exception, LINE 1
- Cannot access private member, LINE 3
- Unresolved external, None – Prints Exception

### Question 3

Correct

Mark 1.00 out of  
1.00

Flag question

Consider the following program and identify the correct output.

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

class Base {
public:
    void f() { cout << "B::f() "; }
};

class Derived : public Base {
public:
    using Base::f;

    void f() { cout << "D::f() "; }
};

int main() {
    Derived d;
    Derived& rd = d;

    d.f();
    rd.f();

    return 0;
}
```

Select one:

- ☐ B::f() B::f()
- ☒ D::f() D::f() ✓
- ☐ B::f() D::f()
- ☐ D::f() B::f()

Your answer is correct.

The correct answer is: D::f() D::f()

#### Question 4

Correct

Mark 2.00 out of  
2.00

🚩 Flag question

Consider the following program.

```

#include <iostream>
using namespace std;

class Base {
protected:
    mutable int value_;
    Base(int value) : value_(value) {}
    virtual ~Base() = 0;
public:
    virtual void f() const = 0;
};

Base::~~Base() { }
void Base::f() const { ++value_; }

template<typename T>
class DerivedType : public Base {
    DerivedType() : Base(sVal) {}
public:
    virtual void f() const;

    static const DerivedType& Type() {
        static DerivedType theObj;

        return theObj;
    }

    static const int sVal;
};

struct T1 {};
struct T2 {};

typedef DerivedType<T1> Type1;
typedef DerivedType<T2> Type2;

template<>
void Type1::f() const { cout << sVal * value_ << " "; Base::f(); }
template<>
void Type2::f() const { Base::f(); cout << sVal * value_ << " "; }

template<>const int Type1::sVal = 2;
template<>const int Type2::sVal = 3;

int main() {
    const Base& r1 = Type1::Type();
    const Base& r2 = Type2::Type();

    r1.f();
    r2.f();
}

```

```

Type1::Type().f();
Type2::Type().f();

return 0;
}

```

What will the output be:

Select one:

- ☐ 4 9 6 12
- ☐ 6 12 8 15
- ☐ 6 9 8 12
- ☒ 4 12 6 15 ✓

Your answer is correct.

The correct answer is: 4 12 6 15

### Question 5

Correct

Mark 1.00 out of 1.00

Flag question

Consider the following program. It uses a wrapper class **IntWrap** for **int**. Add is a template function that should be able to add two **IntWrap** objects. For this, some operator/s in **IntWrap** need/s to be overloaded. Four such candidates are shown commented below.

```

#include <iostream>
using namespace std;

template<typename T> T Add(T a, T b) { return a + b; }

class IntWrap {
    int i_;
public: IntWrap(int n = 0) : i_(n) {}
    //friend IntWrap operator+(IntWrap& a, IntWrap& b);
    //IntWrap operator+(IntWrap& a);
    //IntWrap operator+(int a);
    //operator int();
};

int main() {
    IntWrap i(5);
    IntWrap j(6);
    IntWrap k = Add(i, j);

    return 0;
}

```

If you are allowed to un-comment **only one** operator in the **IntWrap** class and make the code compile and run properly, which is the one that you will **NOT** choose?

Select one:



`IntWrap operator+(int a);`



`friend IntWrap operator+(IntWrap& a, IntWrap& b);`



`operator int();`



`IntWrap operator+(IntWrap& a);`

Your answer is correct.


The correct answer is:

`IntWrap operator+(int a);`

### Question 6

Correct

Mark 1.00 out of  
1.00

 Flag question

Consider the following program and identify the correct output.

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

class Base {
public:
    void f() { cout << "B::f() "; }
};

class Derived : public Base {
public:
    using Base::f;

    void f() const { cout << "D::f() "; }
};

int main() {
    Derived d;
    const Derived& rd = d;

    d.f();
    rd.f();

    return 0;
}
```

Select one:

- ☐ B::f() B::f()
- ☐ D::f() B::f()
- ☐ D::f() D::f()
- ☒ B::f() D::f() ✓

Your answer is correct.

The correct answer is: B::f() D::f()

### Question 7

Correct

Mark 1.00 out of  
1.00

🚩 Flag question

Consider the following program and identify the correct output.

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

class Base {
public:
    virtual void f() { cout << "B::f() "; }
};

class Derived : public Base {
public:
    using Base::f;

    void f() { cout << "D::f() "; }
};

int main() {
    Derived d;
    Derived& rd = d;

    d.Base::f();
    rd.Base::f();

    return 0;
}
```

Select one:

- ☐ B::f() D::f()
- ☐ D::f() D::f()
- ☐ D::f() B::f()
- ☒ B::f() B::f() ✓

Your answer is correct.




The correct answer is: B::f() B::f()

**Question 8**

Correct

Mark 2.00 out of  
2.00

 Flag question

What will be the outcome for the following program?

Note: Line numbers (nn:) are used for reference in output and not part of the program code.

```

01: #include <iostream>
02: #include <exception>
03: #include <string>
04: using namespace std;
05:
06: #define EXCEPTION MyException(__LINE__, __FUNCTION__) \\ S
    ource Line & Function macros
07:
08: class MyException : public exception {
09:     static int E_ID;
10:     int sourceLine_;
11:     string throwingFunction_;
12:     int exceptionID_;
13:     string msg_;
14: public:
15:     MyException(int line, string func) :
16:         sourceLine_(line), throwingFunction_(func), except
    ionID_(E_ID++) { }
17:     const char* what() {
18:         msg_ = "Exception = " + to_string(exceptionID_) +
19:             " at Line = " + to_string(sourceLine_) +
20:             " in Function: " + throwingFunction_;
21:         return msg_.c_str();
22:     }
23: };
24:
25: int MyException::E_ID = 1;
26:
27: void h() {
28:     throw EXCEPTION;
29: }
30: void g() {
31:     try { h(); }
32:     catch (MyException& ex) {
33:         cout << string(ex.what()) << endl;
34:         throw;
35:     }
36: }
37: void f() {
38:     try { g(); }
39:     catch (MyException& ex) {
40:         cout << string(ex.what()) << endl;
41:         throw EXCEPTION;
42:     }
43: }
44: int main() {
45:     try { f(); }
46:     catch (MyException& ex) {
47:         cout << string(ex.what()) << endl;
48:     }
49:     return 0;
50: }

```

Select one:

- ☐ main() aborts
- ☐ Exception = 1 at Line = 28 in Function: h  
Exception = 1 at Line = 28 in Function: h  
Exception = 1 at Line = 28 in Function: h
- ☒ Exception = 1 at Line = 28 in Function: h  
Exception = 1 at Line = 28 in Function: h  
Exception = 2 at Line = 41 in Function: f ✓
- ☐ Exception = 1 at Line = 28 in Function: h  
Exception = 2 at Line = 34 in Function: g  
Exception = 3 at Line = 41 in Function: f

Your answer is correct.

The correct answer is: Exception = 1 at Line = 28 in Function: h  
Exception = 1 at Line = 28 in Function: h  
Exception = 2 at Line = 41 in Function: f

### Question 9

Correct

Mark 1.00 out of  
1.00

Flag question

Consider the following program with three blanks.

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

class Value {
    int sum_;
    int val_;
public:
    Value() : sum_(0), val_(0) {}
    int operator()() {
        sum_ += -----Blank 1----- ;
        return sum_;
    }
};

int main() {
    vector<int> V(5);
    generate(-----Blank 2----- , -----Blank 3----- , Value());
    for (vector<int>::value_type x : V)
        cout << x << " ";

    return 0;
}
```

Choose the correct set to fill in the blanks appropriately so that the output of the program becomes

Select one:



Blank 1: `val_++`  
Blank 2: `V.rbegin()`  
Blank 3: `V.rend()`



Blank 1: `++val_`  
Blank 2: `V.rbegin()`  
Blank 3: `V.rend()`



Blank 1: `++val_`  
Blank 2: `V.begin()`  
Blank 3: `V.end()`



Blank 1: `val_++`  
Blank 2: `V.begin()`  
Blank 3: `V.end()`

Your answer is correct.


The correct answer is:

Blank 1: `val_++`  
Blank 2: `V.rbegin()`  
Blank 3: `V.rend()`

### Question 10

Correct

Mark 1.00 out of  
1.00

 Flag question

Consider the following program.

```

#include <iostream>
using namespace std;

class A {
public:
    virtual ~A() { }
};

class B : public A { };

class C : public B { };

int main() {
    B obj;
    A* p = &obj;
    A& r = obj;

    if (dynamic_cast<A*>(p))
        cout << "A Object" << endl;
    else
        cout << "Not A Object" << endl;

    if (dynamic_cast<B*>(p))
        cout << "B Object" << endl;
    else
        cout << "Not B Object" << endl;

    if (dynamic_cast<C*>(p))
        cout << "C Object" << endl;
    else
        cout << "Not C Object" << endl;

    try {
        A& ra = dynamic_cast<A&>(r);
        cout << "A Object" << endl;

        B& rb = dynamic_cast<B&>(r);
        cout << "B Object" << endl;

        C& rc = dynamic_cast<C&>(r);
        cout << "C Object" << endl;
    }
    catch (bad_cast&) {
        cout << "Not an Object" << endl;
    }

    return 0;
}

```

What will the outcome be:

Select one:

☐ Prints:

```
A Object
Not B Object
Not C Object
A Object
Not an Object
```

☒ Prints:

```
A Object
B Object
Not C Object
A Object
B Object
Not an Object
```



☐ Prints:

```
A Object
B Object
C Object
A Object
B Object
C Object
```

☐ **main()** will abort and program crashes

Your answer is correct.


The correct answer is: Prints:

```
A Object
B Object
Not C Object
A Object
B Object
Not an Object
```

### Question 11

Correct

Mark 1.00 out of  
1.00

 Flag question

Consider the program below.

```

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

class Base {
public:
    virtual void f() { cout << "B::f() "; }
    void g() { cout << "B::g() "; }
};

class Derived : public Base {
public:
    void f() { cout << "D::f() "; }
    void g() { cout << "D::g() "; }
};

int main() {
    Base *rb = new Base();
    Base *rd = new Derived();

    rb->f();
    rb->g();
    rd->f();
    rd->g();

    return 0;
}

```

Match the member function calls with the output from the call:

Member  
call

- (a) **rb.f()**
- (b) **rb.g()**
- (c) **rd.f()**
- (d) **rd.g()**

Call  
Output

- (1) **B::f()**
- (2) **B::g()**
- (3) **D::f()**
- (4) **D::g()**

rd.f()   

rb.g() B::g() ✓

rd.g() B::g() ✓

rb.f() B::f() ✓

Your answer is correct.

The correct answer is: rd.f()

- D::f(), rb.g()
- B::g(), rd.g()
- B::g(), rb.f()
- B::f()

### Question 12

Correct

Mark 2.00 out of  
2.00

Flag question

Consider the throw and catch of exceptions.



```

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

int main() {
    try {
        //throw domain_error("domain_error");    // Throw Line-
1
        //throw 127;                                // Throw Line-
2
        //throw bad_cast();                        // Throw Line-
3
        //throw range_error("range_error");        // Throw Line-
4
    }
    catch (logic_error&) {                        // Catch Line-
1
        cout << "caught logic_error" << endl;
    }
    catch (runtime_error&) {                    // Catch Line-
2
        cout << "caught runtime_error" << endl;
    }
    catch (exception&) {                        // Catch Line-
3
        cout << "caught exception" << endl;
    }
    catch (...) {                                // Catch Line-
4
        cout << "default" << endl;
    }
    cout << "end of program";

    return 0;
}

```

Assume that only one Throw Line is un-commented at a time and match the Throw Line with the Catch Line

Throw line

- (a) Throw Line-1
- (b) Throw Line-2
- (c) Throw Line-3
- (d) Throw Line-4

Catch Line

- (1) Catch Line-1
- (2) Catch Line-2

(1) Catch Line-1

(3) Catch Line-3

(4) Catch Line-4

Throw Line-1  ✓

Throw Line-4  ✓

Throw Line-2  ✓

Throw Line-3  ✓

Your answer is correct.

The correct answer is: Throw Line-1

– Catch Line-1, Throw Line-4

– Catch Line-2, Throw Line-2

– Catch Line-4, Throw Line-3

– Catch Line-3

Finish review

## QUIZ NAVIGATION

**1** 2 3 4 5 6 7 8 9 10 11 12

Show one page at a time

Finish review