

[Quiz](#) >

Review answers

Start date: 9 minutes ago

Complete date: A moment ago

Question 1: Which statement is false about friends?

- ☒ A class cannot access the private members of its friend classes.
- ☐ Friend functions can access the private members of the class they are friend of.
- ☐ Friends violate the information hiding principle of object-oriented programming.
- ☐ Friend functions can access the 'this' pointer.

Question 2: Which statement is true about the following code?

C++:

```
1 | using namespace A::B;
```

- ☒ After this statement, class *B* in namespace *A* can be used without specifying the namespace name.
- ☐ This statement cannot appear inside functions, only at the beginning of your file before any functions.
- ☐ After this statement, classes in the specified namespace cannot be referenced anymore using their full namespace name.
- ☐ After this statement all classes in the namespace *A::B* can be used without specifying the namespace name.

Question 3: What statement is true about the following code?

C++:

```
1 | A::A::B::B() { }
```

- ☐ This code implements function *B* of class *B* that is in namespace *A* that is nested in another namespace *A*.

- ☒ This code does not compile because you can't have two nested namespaces both called A.
- ☐ This code does not compile because member functions in a namespace must be in a `namespace {}` block.
- ☐ This code implements the default constructor of class B that is in namespace A that is nested in another namespace A.

Question 4: Which of the options below is the best operator declaration to add and assign two objects of type `Complex` (`Complex+=Complex`)?

- ☐ `Complex& operator += (const Complex& c) const;`
- ☒ `Complex& operator += (const Complex& c);`
- ☐ `void operator += (const Complex& c);`
- ☐ `Complex operator += (const Complex& c);`

Question 5: Which statement is true about the following code?

C++:

```
1 | delete[] x;
```

- ☐ The code is wrong because the `[]` are missing the size to delete.
- ☐ The code is wrong because the `[]` are not supported with delete.
- ☐ It deallocates the first element of an array pointed by variable x.
- ☒ It deallocates an array pointed by variable x.

Question 6: What is the output of the following program?

C++:

```
1 | int x=10;
2 |
3 | namespace
4 | {
5 |     int x=20;
6 | }
7 |
8 | int main(int x, char* y[])
9 | {
10 |     {
11 |         int x=30;
12 |         std::cout<<:x<<std::endl;
13 |     }
14 |     return 0;
15 | }
```

☐ The number of arguments passed to the program.

☒ 10

☐ 20

Question 7: Which of the operator declarations below is the best way to support the index operator for integer indices (*int*)?

☐ `const Type& operator [] (int index) const; Type operator [] (int index);`

☒ `const Type& operator [] (int index) const; Type& operator [] (int index);`

☐ `Type& operator [] (int index) const;`

☐ `Type operator [] (int index) const;`

Question 8: What is the output of the following code?

C++:

```
1  int size=3; int* a=new int[size];
2  for (int i=0; i<size; i++) a[i]=10-i;
3  std::cout<<a[1]<<" " <<*a<<" " <<(a+1)[0]<<" " <<*a+1<<
4  delete[] a;
```

☐ 9, 10, 9, 9

☐ 9, [address of variable a], 9, [address of variable a + sizeof(int)]

☐ 9, 10, 11, 9

☒ 9, 10, 9, 11

Question 9: Which of the following statements is true about creating a copy constructor and assignment operator?

☒ We need to create a copy constructor and assignment operator because the automatically generated copy constructor and assignment operator copy the data wrongly in certain situations.

☐ We do not need to create a copy constructor and assignment operator because the automatically generated copy constructor and assignment operator do already a member copy.

☐ We need to create a copy constructor and assignment operator because the automatically generated copy constructor and assignment operator do nothing.

☐ We need to create a copy constructor and assignment operator because the "canonical header file rules" dictates it.

Question 10:

Which statement is false about namespaces?

- ☐ Namespaces can be nested.
- ☒ A namespace must be compiled in its own *.lib* file.
- ☐ Namespaces can prevent name collisions or be used to group functionality in logical blocks.
- ☐ Multiple namespace blocks with the same name are possible.

Score: 7 (70.00%)

Pass/Fail: Failed

[Quiz](#) >[Contact us](#) [Advertise](#) [Terms and rules](#) [Privacy policy](#) [Help](#) [Home](#) 

© QUANTNET INC