Question 1

What is the output of the following program?

Code (C++):

1. class A
2. {
3. private:
4. virtual int GetValue() { return 10; }
6. public:
7. int Calculate() { return GetValue()\*1.5; }
8. };
10. class B: public A
11. {
12. private:
13. virtual int GetValue() { return 20; }
14. };
16. int main()
17. {
18. A a; B b;
19. std::cout<<a.Calculate()<<", "<<b.Calculate()<<std::endl;
21. return 0;
22. }

1.  15, 15
2.  This program does not compile.
3.  15, 30
4.  30, 30
5. You answered this question correctly.

Question 2

What is the output of the following program?

Code (C++):

1. class A
2. {
3. private:
4. int m\_data;
6. public:
7. A(): m\_data(GetValue()) {}
8. int Calculate() { return m\_data\*1.5; }
9. virtual int GetValue() { return 10; }
10. };
12. class B: public A
13. {
14. public:
15. virtual int GetValue() { return 20; }
16. };
18. int main()
19. {
20. A a; B b; A\* ap;
22. ap=&a; std::cout<<ap->GetValue()<<", "<<ap->Calculate()<<", ";
23. ap=&b; std::cout<<ap->GetValue()<<", "<<ap->Calculate()<<std::endl;
25. return 0;
26. }

1.  This code does not compile.
2.  10, 15, 20, 30
3.  10, 15, 10, 15
4.  10, 15, 20, 15
5. You answered this question correctly.

Question 3

Which statement is true about aggregation (open diamond shape in UML) and composition (filled diamond shape in UML)

1.  There is no difference between aggregation and composition.
2.  With composition the lifetime of the embedded object is the same as the 'outer' object while with aggregation the lifetime can be different.
3.  With aggregation the lifetime of the embedded object is the same as the 'outer' object while with composition the lifetime can be different.
4.  Aggregation embeds multiple objects while composition only embeds one object.
5. You answered this question correctly.

Question 4

Which statement is false about aggregation?

1.  Aggregation is often combined with delegation to delegate functionality to another object.
2.  Aggregation can hide the functionality of the aggregated object.
3.  Aggregation is a special kind of inheritance.
4.  With aggregation you create an object that consists of one or more other objects.

Question 5

What statement is false about the protected specifier?

1.  Protected members of a base class are accessible from a derived class of a derived class.
2.  Protected must be used on data members of the base class so they can be initialised from a derived class.
3.  Protected member functions should be used if the function is only intended for use by a derived class.
4.  Protected data members should be avoided because it violates the data hiding principle.

Question 6

What is the output of the following program?

Code (C++):

1. class A
2. {
3. private:
4. int m\_da;
6. public:
7. A(int da): m\_da(da) {}
8. int GetA() { return m\_da; }
9. A& operator = (const A& source)
10. { m\_da=source.m\_da; return \*this; }
11. };
13. class B: public A
14. {
15. private:
16. int m\_db;
18. public:
19. B(int da, int db): A(da), m\_db(db) {}
20. int GetB() { return m\_db; }
21. B& operator = (const B& source)
22. { m\_db=source.m\_db; return \*this; }
23. };
25. int main()
26. {
27. B b1(10, 20); B b2(30, 40);
28. b1=b2;
29. std::cout<<b1.GetA()<<", "<<b1.GetB()<<std::endl;
30. return 0;
31. }

1.  10, 20
2.  30, 40
3.  10, 40
4.  30, 20
5. You answered this question correctly.

Question 7

Which statement is false about an abstract class?

1.  Abstract classes cannot be instantiated.
2.  Abstract classes can contain member data.
3.  Abstract classes can only have function declarations, no function implementations.
4.  Abstract classes enable us to work uniformly with a set of related classes.

Question 8

Which statement is false about an abstract class?

1.  Abstract classes enable us to work uniformly with a set of related classes.
2.  Abstract classes can contain member data.
3.  Abstract classes can only have function declarations, no function implementations.
4.  Abstract classes cannot be instantiated.

Question 9

Which statement is false about interfaces?

1.  Interfaces can be used instead of inheritance when there is no 'ISA' relation.
2.  A function working with an interface can work with any set of unrelated classes as long as they implement that interface.
3.  There is not interface construct in C++ but they can be emulated using abstract classes/functions.
4.  Interfaces are allowed to provide default implementations.
5. You answered this question correctly.

Question 10

What statement is true about inheriting operators?

1.  Operators from the base class cannot be overridden in the derived class.
2.  All operators are inherited.
3.  The equal compare operator is inherited but it will only compare the base class data.
4.  When you don't create an assignment operator in the derived class, assigning two derived class objects will only copy the base class data.
5. You answered this question correctly.

Result

Score:

6 correct answers out of 10 questions. You need at least 8 correct answers to pass this quiz.

Pass/Fail:

Failed ([Take this quiz again!](https://www.quantnet.com/quiz/17/take))

Question 1

Which statement is false about an abstract class?

1.  Abstract classes enable us to work uniformly with a set of related classes.
2.  Abstract classes can only have function declarations, no function implementations.
3.  Abstract classes can contain member data.
4.  Abstract classes cannot be instantiated.

Question 2

Which statement is true about aggregation (open diamond shape in UML) and composition (filled diamond shape in UML)

1.  Aggregation embeds multiple objects while composition only embeds one object.
2.  With aggregation the lifetime of the embedded object is the same as the 'outer' object while with composition the lifetime can be different.
3.  There is no difference between aggregation and composition.
4.  With composition the lifetime of the embedded object is the same as the 'outer' object while with aggregation the lifetime can be different.

Question 3

What statement is true about inheriting operators?

1.  The equal compare operator is inherited but it will only compare the base class data.
2.  Operators from the base class cannot be overridden in the derived class.
3.  All operators are inherited.
4.  When you don't create an assignment operator in the derived class, assigning two derived class objects will only copy the base class data.
5. You answered this question correctly.

Question 4

Which statement is false about the colon syntax?

1.  The colon syntax is just another way to initialise your data members. No performance advantages.
2.  The colon syntax must be used to initialise reference data members.
3.  The colon syntax is more efficient when initialising complex data members.
4.  The colon syntax must be used to initialise const data members.
5. You answered this question correctly.

Question 5

Which statement is false about aggregation?

1.  Aggregation can hide the functionality of the aggregated object.
2.  Aggregation is often combined with delegation to delegate functionality to another object.
3.  Aggregation is a special kind of inheritance.
4.  With aggregation you create an object that consists of one or more other objects.

Question 6

What is the output of the following program?

Code (C++):

1. class A
2. {
3. private:
4. virtual int GetValue() { return 10; }
6. public:
7. int Calculate() { return GetValue()\*1.5; }
8. };
10. class B: public A
11. {
12. private:
13. virtual int GetValue() { return 20; }
14. };
16. int main()
17. {
18. A a; B b;
19. std::cout<<a.Calculate()<<", "<<b.Calculate()<<std::endl;
21. return 0;
22. }

1.  This program does not compile.
2.  30, 30
3.  15, 15
4.  15, 30
5. You answered this question correctly.

Question 7

Which statement is false about polymorphism?

1.  Polymorphism is automatically turned on in C++.
2.  Polymorphism only works with pointer or reference variables.
3.  When calling a function via reference/pointer to a base class object, polymorphism enables variation of behaviour depending on the actual derived class object stored in the variable.
4.  When a derived class object is stored in a base class variable, the derived class object functions are hidden.

Question 8

Which statement is false about interfaces?

1.  A function working with an interface can work with any set of unrelated classes as long as they implement that interface.
2.  Interfaces can be used instead of inheritance when there is no 'ISA' relation.
3.  Interfaces are allowed to provide default implementations.
4.  There is not interface construct in C++ but they can be emulated using abstract classes/functions.

Question 9

What is the output of the following program?

Code (C++):

1. class A
2. {
3. private:
4. int m\_data;
6. public:
7. A(): m\_data(GetValue()) {}
8. int Calculate() { return m\_data\*1.5; }
9. virtual int GetValue() { return 10; }
10. };
12. class B: public A
13. {
14. public:
15. virtual int GetValue() { return 20; }
16. };
18. int main()
19. {
20. A a; B b; A\* ap;
22. ap=&a; std::cout<<ap->GetValue()<<", "<<ap->Calculate()<<", ";
23. ap=&b; std::cout<<ap->GetValue()<<", "<<ap->Calculate()<<std::endl;
25. return 0;
26. }

1.  10, 15, 10, 15
2.  This code does not compile.
3.  10, 15, 20, 30
4.  10, 15, 20, 15

Question 10

Which statement is true about inheritance?

1.  All members of the base class are inherited except the constructors, destructor and assignment operator.
2.  All member of the base class are inherited.
3.  Friends of the base class are also friends of the derived class.
4.  Operator functions of the base class are not inherited.

Question 7

What is the output of the following program?

Code (C++):

1. class A
2. {
3. public:
4. std::string F() { return "A"; }
5. };
7. class B: public A
8. {
9. public:
10. std::string F() { return "B"; }
11. };
13. int main()
14. {
15. A a; B b;
16. A\* ap;
18. std::cout<<a.F()<<", ";
19. std::cout<<b.F()<<", ";
21. ap=&a; std::cout<<ap->F()<<", ";
22. ap=&b; std::cout<<ap->F()<<std::endl;
24. return 0;
25. }

1.  A, A, A, A
2.  A, B, A, A
3.  This code does not compile.
4.  A, B, A, B
5. You answered this question correctly.

Question 8

What is the output of the following program?

Code (C++):

1. class A
2. {
3. public:
4. virtual std::string F() = 0;
5. };
7. class B: public A
8. {
9. public:
10. std::string F() { return "B"; }
11. };
13. int main()
14. {
15. A a; B b;
16. A\* ap=&b;
18. std::cout<<b.F()<<", ";
19. std::cout<<ap->F()<<std::endl;
21. return 0;
22. }

1.  This code does not compile.
2.  B, 0
3.  B,
4.  B, B