# C++ Programming Questions and Answers – Linkage

1.Where is the derived class is derived from?  
a) derived  
b) base  
c) both derived & base  
d) class

Answer: b  
Explanation: Because derived inherits functions and variables from base.

2. Pick out the correct statement.  
a) A derived class’s constructor cannot explicitly invokes its base class’s constructor  
b) A derived class’s destructor cannot invoke its base class’s destructor  
c) A derived class’s destructor can invoke its base class’s destructor  
d) A derived class’s destructor can invoke its base & derived class’s destructor  
Answer: b  
Explanation: Destructors are automatically invoked when an object goes out of scope or when a dynamically allocated object is deleted. Inheritance does not change this behavior. This is the reason a derived destructor cannot invoke its base class destructor.

3. Which of the following can derived class inherit?  
a) members  
b) functions  
c) both members & functions  
d) classes  
Answer: c  
Explanation: Both data members and member functions are inherited by derived class in C++.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class A
4. {
5. public:
6. A(int n )
7. {
8. cout << n;
9. }
10. };
11. class B: public A
12. {
13. public:
14. B(int n, double d)
15. : A(n)
16. {
17. cout << d;
18. }
19. };
20. class C: public B
21. {
22. public:
23. C(int n, double d, char ch)
24. : B(n, d)
25. {
26. cout <<ch;
27. }
28. };
29. int main()
30. {
31. C c(5, 4.3, 'R');
32. return 0;
33. }

a) 54.3R  
b) R4.35  
c) 4.3R5  
d) R2.6

Answer: a  
Explanation: In this program, We are passing the value and manipulating by using the derived class.  
Output:

$ g++ der.cpp

$ a.out

54.3R

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class BaseClass
4. {
5. protected:
6. int i;
7. public:
8. BaseClass(int x)
9. {
10. i = x;
11. }
12. ~BaseClass()
13. {
14. }
15. };
16. class DerivedClass: public BaseClass
17. {
18. int j;
19. public:
20. DerivedClass(int x, int y): BaseClass(y)
21. {
22. j = x;
23. }
24. ~DerivedClass()
25. {
26. }
27. void show()
28. {
29. cout << i << " " << j << endl;
30. }
31. };
32. int main()
33. {
34. DerivedClass ob(3, 4);
35. ob.show();
36. return 0;
37. }

a) 3 4  
b) 4 3

c) 4  
d) 3

Answer: b  
Explanation: In this program, We are passing the values and assigning it to i and j and we are printing it.  
Output:

$ g++ der1.cpp

$ a.out

4 3

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5. public:
6. int m;
7. Base(int n=0)
8. : m(n)
9. {
10. cout << "Base" << endl;
11. }
12. };
13. class Derived: public Base
14. {
15. public:
16. double d;
17. Derived(double de = 0.0)
18. : d(de)
19. {
20. cout << "Derived" << endl;
21. }
22. };
23. int main()
24. {
25. cout << "Instantiating Base" << endl;
26. Base cBase;
27. cout << "Instantiating Derived" << endl;
28. Derived cDerived;
29. return 0;
30. }

a)

Instantiating Base

Base

Instantiating Derived

Base

Derived

b)

Instantiating Base

Instantiating Derived

Base

Derived

c)

Instantiating Base

Base

Instantiating Derived

Base

d) Instantiating Base

Answer: a  
Explanation: In this program, We are printing the execution order of the program.  
Output:

$ g++ der2.cpp

$ a.out

Instantiating Base

Base

Instantiating Derived

Base

Derived

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Parent
4. {
5. public:
6. Parent (void)
7. {
8. cout << "Parent()**\n**";
9. }
10. Parent (int i)
11. {
12. cout << "Parent("<< i << ")**\n**";
13. };
14. Parent (void)
15. {
16. cout << "~Parent()**\n**";
17. };
18. };
19. class Child1 : public Parent { };
20. class Child2 : public Parent
21. {
22. public:
23. Child2 (void)
24. {
25. cout << "Child2()**\n**";
26. }
27. Child2 (int i) : Parent (i)
28. {
29. cout << "Child2(" << i << ")**\n**";
30. }
31. ~Child2 (void)
32. {
33. cout << "~Child2()**\n**";
34. }
35. };
36. int main (void)
37. {
38. Child1 a;
39. Child2 b;
40. Child2 c(42);
41. return 0;
42. }

a)

Parent()

Parent()

Child2()

Parent(42)

Child2(42)

~Child2()

~Parent()

~Child2()

~Parent()

~Parent()

b) error  
c) runtime error  
d) Parent(42)

Answer: b  
Explanation: In this program, We got an error in overloading because we didn’t invoke the destructor of parent.

8. What will be the output of the following C++ code?

1. #include<iostream>
2. using namespace std;
3. class X
4. {
5. int m;
6. public:
7. X() : m(10)
8. {
9. }
10. X(int mm): m(mm)
11. {
12. }
13. int getm()
14. {
15. return m;
16. }
17. };
18. class Y : public X
19. {
20. int n;
21. public:
22. Y(int nn) : n(nn) {}
23. int getn() { return n; }
24. };
25. int main()
26. {
27. Y yobj( 100 );
28. cout << yobj.getm() << " " << yobj.getn() << endl;
29. }

a) 10 100  
b) 100 10  
c) 10 10  
d) 100 100

Answer: a  
Explanation: In this program, We are passing the value and getting the result by derived class.  
Output:

$ g++ der5.cpp

$ a.out

10 100

9. Which operator is used to declare the destructor?  
a) #  
b) ~  
c) @  
d) $

Answer: b  
Explanation: tilde(~) is used to declare destructor of a class.

10. Which constructor will initialize the base class data member?  
a) derived class  
b) base class  
c) class  
d) derived & base class

Answer: b  
Explanation: Because it is having the proper data set to initialize, Otherwise it will throw an error.

# C++ Programming Questions and Answers – Header Files Usage

1. What is the user-defined header file extension in c++?  
a) cpp  
b) h  
c) hf  
d) hg

Answer: b  
Explanation: .h extensions are used for user defined header files. To include a user defined header file one should use #include”name.h” i.e. enclosed within double quotes.

2. Which of the following keyword is used to declare the header file?  
a) include  
b) exclude  
c) string  
d) namespace

Answer: a  
Explanation: The include keyword is used to include all the required things to execute the given code in the program.

3. Identify the incorrect statement.  
a) iostream is a standard header and iostream.h is a non-standard header  
b) iostream is a non-standard header and iostream.h is a non-standard header  
c) iostream is a standard header and iostream.h is a standard header  
d) iostream is a non-standard header

Answer: a  
Explanation: The iostream.h is used in the older versions of c++ and iostream is evolved from it in the std namespace.

4. What does a default header file contain?  
a) prototype  
b) implementation  
c) declarations  
d) pointing

Answer: c  
Explanation: In the header file, we define something that to be manipulated in the program.

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. char name[30];
6. cout << "Enter name: ";
7. gets(name);
8. cout << "Name: ";
9. puts(name);
10. return 0;
11. }

a) jobsjobs  
b) jobs  
c) compile time error  
d) program will not run

Answer: c  
Explanation: This program will run on older version of C++ with the inclusion of #include header file, but for on new compiler C++14 and above the gets is removed from the header file so it will not run on them even after inclusion of cstdio header file.

6. setprecision requires which of the following header file?  
a) stdlib.h  
b) iomanip.h  
c) console.h  
d) conio.h

Answer: b  
Explanation: The iomanip header file is used to correct the precision of the values.

7. Which of the following header file does not exist?  
a) <iostream>  
b) <string>  
c) <sstring>  
d) <sstream>

Answer: c  
Explanation: There is no such header file <sstring> in C++.

8. Which of the header file must be included to use stringstream?  
a) <iostream>  
b) <string>  
c) <sstring>  
d) <sstream>

Answer: b  
Explanation: stringstream is available under the header file <string> in C++.

9. Which of the following header files is required for creating and reading data files?  
a) ofstream.h  
b) fstream.h  
c) ifstream.h  
d) console.h

Answer: b  
Explanation: In this fstream.h header file is used for accessing the files only.

10. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <stdarg.h>
3. using namespace std;
4. float avg( int Count, ... )
5. {
6. va\_list Numbers;
7. va\_start(Numbers, Count);
8. int Sum = 0;
9. for (int i = 0; i < Count; ++i)
10. Sum += va\_arg(Numbers, int);
11. va\_end(Numbers);
12. return (Sum/Count);
13. }
14. int main()
15. {
16. float Average = avg(10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
17. cout << Average;
18. return 0;
19. }

a) 4  
b) 5  
c) 6  
d) compile time error

Answer: a  
Explanation: In this program, we are finding the average of first 10 numbers using stdarg header file  
Output:

$ g++ std.cpp

$ a.out 4

# C++ Programming Questions and Answers – Classes – 1

1. What does a class in C++ holds?  
a) data  
b) functions  
c) both data & functions  
d) arrays

Answer: c  
Explanation: The classes in C++ encapsulates(i.e. put together) all the data and functions related to them for manipulation.

2. How many specifiers are present in access specifiers in class?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three types of access specifiers. They are public, protected and private.

3. Which is used to define the member of a class externally?  
a) :  
b) ::  
c) #  
d) !!$

Answer: b  
Explanation: :: operator is used to define the body of any class function outside the class.

4. Which other keywords are also used to declare the class other than class?  
a) struct  
b) union  
c) object  
d) both struct & union

Answer: d  
Explanation: In C++, the other keywords used to declare a class-like structure, apart from class, are:  
struct: This can be considered similar to class.  
union: This can be considered a special class type where all members share the same memory location.

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class rect
4. {
5. int x, y;
6. public:
7. void val (int, int);
8. int area ()
9. {
10. return (x \* y);
11. }
12. };
13. void rect::val (int a, int b)
14. {
15. x = a;
16. y = b;
17. }
18. int main ()
19. {
20. rect rect;
21. rect.val (3, 4);
22. cout << "rect area: " << rect.area();
23. return 0;
24. }

a) rect area: 24  
b) rect area: 12  
c) compile error because rect is as used as class name and variable name in line #20  
d) rect area: 56

Answer: b  
Explanation: In this program, we are calculating the area of rectangle based on given values.  
Output:

$ g++ class.cpp

$ a.out

rect area: 12

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class CDummy
4. {
5. public:
6. int isitme (CDummy& param);
7. };
8. int CDummy::isitme (CDummy& param)
9. {
10. if (&param == this)
11. return true;
12. else
13. return false;
14. }
15. int main ()
16. {
17. CDummy a;
18. CDummy \*b = &a;
19. if (b->isitme(a))
20. {
21. cout << "execute";
22. }
23. else
24. {
25. cout<<"not execute";
26. }
27. return 0;
28. }

a) execute  
b) not execute  
c) error  
d) both execute & not execute

Answer: a  
Explanation: In this program, we are just pointing the pointer to a object and printing execute if it is correctly pointed.  
Output:

$ g++ class1.cpp

$ a.out

execute

7. Which of the following is a valid class declaration?  
a) class A { int x; };  
b) class B { }  
c) public class A { }  
d) object A { int x; };

Answer: a  
Explanation: A class declaration terminates with semicolon and starts with class keyword. only option (a) follows these rules therefore class A { int x; }; is correct.

8. The data members and functions of a class in C++ are by default \_\_\_\_\_\_\_\_\_\_\_\_  
a) protected  
b) private  
c) public  
d) public & protected

Answer: b  
Explanation: By default all the data members and member functions of class are private.

9. Constructors are used to \_\_\_\_\_\_\_\_\_\_\_\_  
a) initialize the objects  
b) construct the data members  
c) both initialize the objects & construct the data members  
d) delete the objects

Answer: a  
Explanation: Once the object is declared means, the constructor are also declared by default.

10. When struct is used instead of the keyword class means, what will happen in the program?  
a) access is public by default  
b) access is private by default  
c) access is protected by default  
d) access is denied

Answer: a  
Explanation: For structures, by default all the data members and member functions are public.

**C++ Programming Questions and Answers – Classes – 2**

1. Which category of data type a class belongs to?  
a) Fundamental data type  
b) Derived data type  
c) User defined derived data type  
d) Atomic data type

Answer: c  
Explanation: Fundamental/Atomic data type includes int, char, float, double and void. Derived data type includes arrays, pointers, references, function and constants. User defined derived data type includes class, structure, union and enumeration.

2. Which operator a pointer object of a class uses to access its data members and member functions?  
a) .  
b) ->  
c) :  
d) ::

Answer: b  
Explanation: ->(arrow operator) is used by a pointer object to access members of its class.

3. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a;

public:

int assign(int i) const {

a = i;

}

int return\_value() const {

return a;

}

};

int main(int argc, char const \*argv[])

{

A obj;

obj.assign(5);

cout<<obj.return\_value();

}

a) 5  
b) 10  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: As the assign() is a constant function and a constant function cannot change the state of an object and as in the assign function we are trying to modify the member a of the object therefore the program gives error.

4. What is the correct syntax of accessing a static member of a Class?

---------------------------

Example class:

class A

{

public:

static int value;

}

---------------------------

a) A.value  
b) A::value  
c) A->value  
d) A^value

Answer: b  
Explanation: Scope resolution operator(::) is used to access a static member of a class.

5. How the objects are self-referenced in a member function of that class.  
a) Using a special keyword object  
b) Using this pointer  
c) Using \* with the name of that object  
d) By passing self as a parameter in the member function

Answer: b  
Explanation: In Classes objects are self-referenced using this pointer inside the member functions. for example this->value to access the data member value of that object.

6. What does a mutable member of a class mean?  
a) A member that can never be changed  
b) A member that can be updated only if it not a member of constant object  
c) A member that can be updated even if it a member of constant object  
d) A member that is global throughout the class

Answer: c  
Explanation: Mutable members are those which can be updated even if it a member of a constant object. You can change their value even from a constant member function of that class.

7. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

mutable int a;

public:

int assign(int i) const {

a = i;

}

int return\_value() const {

return a;

}

};

int main(int argc, char const \*argv[])

{

A obj;

obj.assign(5);

cout<<obj.return\_value();

}

a) 5  
b) Error  
c) Segmentation fault  
d) Undefined value

Answer: a  
Explanation: As a is mutable member of the class it’s value can be modified whether it is a part of constant object or not. It can be modified even inside a constant member function. Hence, the program tuns fine and does not gives any error.

8. Pick the incorrect statement about inline functions in C++?  
a) They reduce function call overheads  
b) These functions are inserted/substituted at the point of call  
c) Saves overhead of a return call from a function  
d) They are generally very large and complicated function

Answer: d  
Explanation: Inline are functions that are expanded when it is called. The whole code of the inline function gets inserted/substituted at the point of call. In this, they help in reducing the function call overheads. Also they save overhead of a return call from a function. Inline functions are generally kept small.

9. Inline functions are avoided when \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) function contains static variables  
b) function have recursive calls  
c) function have loops  
d) all of the mentioned

Answer: d  
Explanation: Inline functions are avoided in all the above cases as whole inline code is copied to the point of call so compiler avoids to make large functions as inline. Even if you yourself mention inline but the function is large compiler ignores your request of inline and treats that function as a normal function.

10. Pick the correct statement.  
a) Macros and inline functions are same thing  
b) Macros looks like function calls but they are actually not  
c) Inline functions looks like function but they are not  
d) Inline function are always large

Answer: b  
Explanation: Macros in C++ looks like function calls but actually they are not function calls.

11. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

class S

{

int m;

public:

#define MAC(S::m)

};

int main(int argc, char const \*argv[])

{

cout<<"Hello World";

return 0;

}

a) Hello World  
b) Error  
c) Segmentation Fault  
d) Blank Space

Answer: b  
Explanation: Macros cannot access the private member of a class therefore #define MAC(S::m) will give an error.

12. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

static int a;

public:

void change(int i){

a = i;

}

void value\_of\_a(){

cout<<a;

}

};

int main(int argc, char const \*argv[])

{

A a1 = A();

a1.change(5);

a1.value\_of\_a();

return 0;

}

a) 5  
b) Garbage value  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: Every static member of a class is initialised before its use. As ‘a’ is a static member of the class and is not initialised so the program will give error.

13. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

static int a;

public:

void change(int i){

a = i;

}

void value\_of\_a(){

cout<<a;

}

};

int A::a = 5;

int main(int argc, char const \*argv[])

{

A a1 = A();

A a2 = A();

A a3 = A();

a1.change(10);

a1.value\_of\_a();

a2.value\_of\_a();

a3.value\_of\_a();

return 0;

}

a) 1055  
b) 555  
c) 101010  
d) 51010

Answer: c  
Explanation: As ‘a’ is a static member of the class so it is a type of global variable to the class i.e. any change made by one object is reflected back to all the other objects. Hence when a is changed to 10 by object a1, so value of ‘a’ becomes 10 for each object and 3 times 10 is printed.

14. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a = 5;

public:

void change(int i){

a = i;

}

static void value\_of\_a(){

cout<<a;

}

};

int main(int argc, char const \*argv[])

{

A a1 = A();

a1.change(10);

a1.value\_of\_a();

return 0;

}

a) 10  
b) Error  
c) Segmentation Fault  
d) 5

Answer: b  
Explanation: As value\_of\_a() is a static function and static member can access only static members therefore the program will give error.

15. Which functions of a class are called inline functions?  
a) All the functions containing declared inside the class  
b) All functions defined inside or with the inline keyword  
c) All the functions accessing static members of the class  
d) All the functions that are defined outside the class  
Answer: b  
Explanation: All the functions defined inside the class or functions having inline keyword before them are inline functions of a class provided they are small and simple otherwise compiler ignores the request of inline.

# C++ Programming Questions and Answers – User Defined Types

1. Which keyword is used to define the user defined data types?  
a) def  
b) union  
c) typedef  
d) type

Answer: c  
Explanation: Typedef is used to define user defined datatypes.  
eg:  
typedef int INT;  
INT a;  
here INT is used defined data type.

2. Identify the correct statement.  
a) typedef does not create different types. It only creates synonyms of existing types  
b) typedef create different types  
c) typedef create own types  
d) typedef will not creates synonyms of existing types

Answer: a  
Explanation: By using typedef, we can create a type of pre-existing type only not our own type of data.

3. What does the data type defined by union will do?  
a) It allow one different portion of memory to be accessed as same data types  
b) It allow one same portion of memory to be accessed as same data types  
c) It allow one different portion of memory to be accessed as different data types  
d) It allow one same portion of memory to be accessed as different data types

Answer: d  
Explanation: Union is used to define the data types of our choice and it will store the data type in one location make them accessible.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. typedef int num;
6. num a = 10, b = 15;
7. num c = a + b + a - b;
8. cout << c;
9. return 0;
10. }

a) 20  
b) 15  
c) 30  
d) 25

Answer: a  
Explanation: In this program, we are manipulating the numbers and printing the result using user-defined data types.  
Output:

$ g++ user.cpp

$ a.out

20

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int i;
6. enum month
7. {
8. JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,DEC
9. };
10. for (i = JAN; i <= DEC; i++)
11. cout << i;
12. return 0;
13. }

a) 012345678910  
b) 0123456789  
c) 01234567891011  
d) 01234567891011122

Answer: a  
Explanation: In this program, we are defined the data types as enumerator and printing its value in a order.  
Output:

$ g++ user1.cpp

$ a.out

012345678910

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. typedef int num;
6. typedef char let;
7. let w = "steve";
8. num a = 10, b = 15;
9. num c = a + w;
10. cout << c;
11. return 0;
12. }

a) 10steve  
b) steve10  
c) compile time error  
d) compile but not run

Answer: c  
Explanation: Error: invalid conversion from ‘const char\*’ to ‘let {aka char}’.

7. What is the syntax of user-defined data types?  
a) typedef ExistingDataType NameByUser  
b) typedef NameByUser ExistingDataType  
c) def NameByUser ExistingDataType  
d) def NameByUser ExistingData

Answer: a  
Explanation: correct syntax is typedef ExistingDataType NameByUser.  
typedef int INT; (typedef existing-datatype New-name;).

8. How many types of user-defined data type are in c++?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three types of user-defined data types. They are typedef, union, enumerator.

9. What is the scope of typedef defined data types?  
a) inside that block only  
b) whole program  
c) outside the program  
d) main function

Answer: b  
Explanation: We are defining the user-defined data type to be availed only inside that program, if we want to use anywhere means we have to define those types in the header file.

10. How many types of models are available to create the user-defined data type?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: There are two types of models. They are references to built-in types and multipart types.

# C++ Programming Questions and Answers – Objects

1. Where does the object is created?  
a) class  
b) constructor  
c) destructor  
d) attributes

Answer: a  
Explanation: In class, only all the listed items except class will be declared.

2. How to access the object in the class?  
a) scope resolution operator  
b) ternary operator  
c) direct member access operator  
d) resolution operator

Answer: c  
Explanation: Objects in the method can be accessed using direct member access operator which is (.).

3. Which of these following members are not accessed by using direct member access operator?  
a) public  
b) private  
c) protected  
d) both private & protected

Answer: d  
Explanation: Because of the access is given to the private and protected, We can’t access them by using direct member access operator.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Box
4. {
5. public :
6. double length;
7. double breadth;
8. double height;
9. };
10. int main( )
11. {
12. Box Box1;
13. double volume;
14. Box1.height = 5;
15. Box1.length = 6;
16. Box1.breadth = 7.1;
17. volume = Box1.height \* Box1.length \* Box1.breadth;
18. cout << "Volume of Box1 : " << volume <<endl;
19. return 0;
20. }

a) 210  
b) 213  
c) 215  
d) 217

Answer: b  
Explanation: In the above program, we are calculating the area of the cube by using the cube formula  
Output:

$ g++ obj1.cpp

$ a.out

213

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Rect
4. {
5. int x, y;
6. public:
7. void set\_values (int,int);
8. int area ()
9. {
10. return (x \* y);
11. }
12. };
13. void Rect::set\_values (int a, int b)
14. {
15. x = a;
16. y = b;
17. }
18. int main ()
19. {
20. Rect recta, rectb;
21. recta.set\_values (5, 6);
22. rectb.set\_values (7, 6);
23. cout << "recta area: " << recta.area();
24. cout << "rectb area: " << rectb.area();
25. return 0;
26. }

a) recta area: 30 rectb area: 42  
b) recta area: 20 rectb area: 34  
c) recta area: 30 rectb area: 21  
d) recta area: 30 rectb area: 33

Answer: a  
Explanation: We are calculating the area of rectangle by two objects.

6. Pick out the other definition of objects.  
a) member of the class  
b) associate of the class  
c) attribute of the class  
d) instance of the class

Answer: d  
Explanation: An Object represents an instance of a class i.e. a variable of that class type having access to its data members and member functions from outside if allowed.

7. How many objects can present in a single class?  
a) 1  
b) 2  
c) 3  
d) as many as possible

Answer: d  
Explanation: Because a class may contain any number of objects according to its compliance.

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample
4. {
5. private:
6. int var;
7. public:
8. void input()
9. {
10. cout << var;
11. }
12. void output()
13. {
14. cout << "Variable entered is ";
15. cout << var << "**\n**";
16. }
17. };
18. int main()
19. {
20. sample object;
21. object.input();
22. object.output();
23. object.var();
24. return 0;
25. }

a)

Enter an integer 5

Variable entered is 5

b) Runtime error  
c) Error  
d)

Enter an integer 7

Variable entered is 7

Answer: c  
Explanation: There is no member function var() in the class hence the program will through an error stating var is a private data member and it cannot be used as a function.

9. Which special character is used to mark the end of class?  
a) ;  
b) :  
c) #  
d) $

Answer: a  
Explanation: Similar to ending any statement, a class is also terminated with semicolon(;).

10. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class number
4. {
5. int i;
6. public:
7. int geti();
8. void puti(int j);
9. };
10. int number::geti()
11. {
12. return i;
13. }
14. void number::puti(int j)
15. {
16. i = j;
17. }
18. int main()
19. {
20. number s;
21. s.puti(10);
22. cout << s.geti( );
23. return 0;
24. }

a) 10  
b) 11  
c) 20  
d) 22

Answer: a  
Explanation: We are getting the number and copying it to j and printing it.  
Output:

$ g++ obj2.cpp

$ a.out

10

# C++ Programming Questions and Answers – Operator Functions

1. Pick the other name of operator function.  
a) function overloading  
b) operator overloading  
c) member overloading  
d) object overloading

Answer: b  
Explanation: Operator function means operation defined for that operator so if user defines a function for an operator then that is called operator overloading i.e. overloading already present operator function.

2. Which of the following operators can’t be overloaded?  
a) ::  
b) +  
c) –  
d) []

Answer: a  
Explanation: :: operator cannot be overloaded because this operator operates on names rather than values and C++ has no syntax for writing codes that works on names than values so using syntax these operators cannot be overloaded.

3. How to declare operator function?  
a) operator sign  
b) operator  
c) name of the operator  
d) name of the class

Answer: a  
Explanation: We have to declare the operator function by using the operator, operator sign. Example “operator +” where the operator is a keyword and + is the symbol need to be overloaded.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample
4. {
5. public:
6. int x, y;
7. sample() {};
8. sample(int, int);
9. sample operator + (sample);
10. };
11. sample::sample (int a, int b)
12. {
13. x = a;
14. y = b;
15. }
16. sample sample::operator+ (sample param)
17. {
18. sample temp;
19. temp.x = x + param.x;
20. temp.y = y + param.y;
21. return (temp);
22. }
23. int main ()
24. {
25. sample a (4,1);
26. sample b (3,2);
27. sample c;
28. c = a + b;
29. cout << c.x << "," << c.y;
30. return 0;
31. }

a) 5, 5  
b) 7, 3  
c) 3, 7  
d) 3, 5

Answer: b  
Explanation: In this program, we are adding the first number of a with first number of b by using operator function and also we are adding second number by this method also.  
Output:

$ g++ oper.cpp

$ a.out

7, 3

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Box
4. {
5. double length;
6. double breadth;
7. double height;
8. public:
9. double getVolume(void)
10. {
11. return length \* breadth \* height;
12. }
13. void setLength( double len )
14. {
15. length = len;
16. }
17. void setBreadth( double bre )
18. {
19. breadth = bre;
20. }
21. void setHeight( double hei )
22. {
23. height = hei;
24. }
25. Box operator+(const Box& b)
26. {
27. Box box;
28. box.length = this->length + b.length;
29. box.breadth = this->breadth + b.breadth;
30. box.height = this->height + b.height;
31. return box;
32. }
33. };
34. int main( )
35. {
36. Box Box1;
37. Box Box2;
38. Box Box3;
39. double volume = 0.0;
40. Box1.setLength(6.0);
41. Box1.setBreadth(7.0);
42. Box1.setHeight(5.0);
43. Box2.setLength(12.0);
44. Box2.setBreadth(13.0);
45. Box2.setHeight(10.0);
46. volume = Box1.getVolume();
47. cout << "Volume of Box1 : " << volume <<endl;
48. volume = Box2.getVolume();
49. cout << "Volume of Box2 : " << volume <<endl;
50. Box3 = Box1 + Box2;
51. volume = Box3.getVolume();
52. cout << "Volume of Box3 : " << volume <<endl;
53. return 0;
54. }

a)

Volume of Box1 : 210

Volume of Box2 : 1560

Volume of Box3 : 5400

b)

Volume of Box1 : 200

Volume of Box2 : 1560

Volume of Box3 : 5400

c)

Volume of Box1 : 210

Volume of Box2 : 1550

Volume of Box3 : 5400

d)

Volume of Box1 : 200

Volume of Box2 : 1000

Volume of Box3 : 5260

Answer: a  
Explanation: In this program, we finding the box3 area by adding box1 and box2.  
Output:

$ g++ oper1.cpp

$ a.out

Volume of Box1 : 210

Volume of Box2 : 1560

Volume of Box3 : 5400

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Integer
4. {
5. int i;
6. public:
7. Integer(int ii) : i(ii) {}
8. const Integer
9. operator+(const Integer& rv) const
10. {
11. cout << "operator+" << endl;
12. return Integer(i + rv.i);
13. }
14. Integer&
15. operator+=(const Integer& rv)
16. {
17. cout << "operator+=" << endl;
18. i += rv.i;
19. return \*this;
20. }
21. };
22. int main()
23. {
24. int i = 1, j = 2, k = 3;
25. k += i + j;
26. Integer ii(1), jj(2), kk(3);
27. kk += ii + jj;
28. }

a)

operator+

operator+=

b)

operator+=

operator+

c)

operator+

operator+

d)

operator+

operator=

Answer: a  
Explanation: We are using two operator functions and executing them and the result is printed according to the order.  
Output:

$ g++ oper2.cpp

$ a.out

operator+

operator+=

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class myclass
4. {
5. public:
6. int i;
7. myclass \*operator->()
8. {return this;}
9. };
10. int main()
11. {
12. myclass ob;
13. ob->i = 10;
14. cout << ob.i << " " << ob->i;
15. return 0;
16. }

a) 10 10  
b) 11 11  
c) error  
d) runtime error

Answer: a  
Explanation: In this program, -> operator is used to describe the member of the class and so we are getting this output.  
Output:

$ g++ char4.cpp

$ a.out

10 10

8. Which of the following statements is NOT valid about operator overloading?  
a) Only existing operators can be overloaded  
b) The overloaded operator must have at least one operand of its class type  
c) The overloaded operators follow the syntax rules of the original operator  
d) None of the mentioned

Answer: b  
Explanation: The overloaded operator must not have at least one operand of its class type.

9. Operator overloading is \_\_\_\_\_\_\_\_\_\_\_  
a) making c++ operator works with objects  
b) giving new meaning to existing operator  
c) making the new operator  
d) adding operation to the existing operators

Answer: d  
Explanation: Operator overloading is the way adding operation to the existing operators.

10. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. ostream & operator<<(ostream & i, int n)
4. {
5. return i;
6. }
7. int main()
8. {
9. cout << 5 << endl;
10. cin.get();
11. return 0;
12. }

a) 5  
b) 6  
c) error  
d) runtime error

Answer: c  
Explanation: In this program, there will arise an ambiguous overload for 5.

**C++ Programming Questions and Answers – Operator Overloading – 1**

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

static int a;

public:

void show()

{

a++;

cout<<"a: "<<a<<endl;

}

};

int A::a = 5;

int main(int argc, char const \*argv[])

{

A a;

return 0;

}

a) Error as a private member a is referenced outside the class  
b) Segmentation fault  
c) No output  
d) Program compiles successfully but gives run-time error

Answer: c  
Explanation: As every static member must be initialized and we have initialized variable ‘a’ so no run time error. Also as variable ‘a’ is a static member and is referenced using the class for initialization therefore no compiler error.

2. What happens when objects s1 and s2 are added?

string s1 = "Hello";

string s2 = "World";

string s3 = (s1+s2).substr(5);

a) Error because s1+s2 will result into string and no string has substr() function  
b) Segmentation fault as two string cannot be added in C++  
c) The statements runs perfectly  
d) Run-time error

Answer: c  
Explanation: string is class in C++, therefore when we do (s1+s2) a temporary object is created which stores the result of s1+s2 and then that object calls the function substr() and as that is an object of string class hence substr is a callable function for that temporary string object.

3. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

static int a;

public:

A()

{

cout<<"Object of A is created**\n**";

}

void show()

{

a++;

cout<<"a: "<<a<<endl;

}

};

class B

{

public:

};

int main(int argc, char const \*argv[])

{

A a1, a2;

A a3 = a1 + a2;

return 0;

}

a) Runs perfectly  
b) Run-time Error  
c) Segmentation fault  
d) Compile-time Error

Answer: d  
Explanation: As the programmer has not defined what action should be taken when two objects of class A are added, so the program doesn’t know and gives compile time error.

4. What is operator overloading in C++?  
a) Overriding the operator meaning by the user defined meaning for user defined data type  
b) Redefining the way operator works for user defined types  
c) Ability to provide the operators with some special meaning for user defined data type  
d) All of the mentioned

Answer: d  
Explanation: Operator overloading helps programmer to give his/her own meaning to an operator for user defined data types(eg, classes).

5. What is the syntax of overloading operator + for class A?  
a) A operator+(argument\_list){}  
b) A operator[+](argument\_list){}  
c) int +(argument\_list){}  
d) int [+](argument\_list){}

Answer: a  
Explanation: The general syntax for operator overloading is:

return\_type operator\_keywordOperator(argument\_list){}  
eg.  
A opeartor+(argument\_list){}

6. How many approaches are used for operator overloading?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are 3 different approaches used for operator overloading:  
i. Overloading unary operator.  
ii. Overloading binary operator.  
iii. Overloading binary operator using a friend function.

7. Which of the following operator cannot be overloaded?  
a) +  
b) ?:  
c) –  
d) %

Answer: b  
Explanation: ?:, :: and . cannot be overloaded +, -, % can be overloaded.

8. Which of the following operator can be overloaded?  
a) ?:  
b) ::  
c) .  
d) ==

Answer: d  
Explanation: ?:, :: and . cannot be overloaded whereas == can be overloaded.

9. Which of the following operator cannot be used to overload when that function is declared as friend function?  
a) -=  
b) ||  
c) ==  
d) []

Answer: d  
Explanation: When an operator overlaoded function is declared as friend function then [] cannot be overloaded.

10. Which of the following operator can be used to overload when that function is declared as friend function?  
a) []  
b) ()  
c) ->  
d) |=

Answer: d  
Explanation: When an operator overlaoded function is declared as friend function then [], () and -> cannot be overloaded.

11. In case of non-static member functions how many maximum object arguments a unary operator overloaded function can take?  
a) 1  
b) 2  
c) 3  
d) 0

Answer: d  
Explanation: In the case of non-static member functions unary operator overloaded function should not take any object argument.

12. In case of non-static member functions how many maximum object arguments a binary operator overloaded function can take?  
a) 1  
b) 2  
c) 3  
d) 0

Answer: a  
Explanation: In the case of non-static member functions binary operator overloaded function should take maximum one object argument only.

13. In the case of friend operator overloaded functions how many maximum object arguments a unary operator overloaded function can take?  
a) 1  
b) 2  
c) 3  
d) 0

Answer: a  
Explanation: In the case of friend operator overloaded functions unary operator overloaded function should take maximum one object argument only.

14. In the case of friend operator overloaded functions how many maximum object arguments a binary operator overloaded function can take?  
a) 1  
b) 2  
c) 3  
d) 0

Answer: b  
Explanation: In the case of friend operator overloaded functions binary operator overloaded function should take maximum two object argument.

15. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

static int a;

public:

void show()

{

a++;

cout<<"a: "<<a<<endl;

}

void operator.()

{

cout<<"Objects are added**\n**";

}

};

class B

{

public:

};

int main(int argc, char const \*argv[])

{

A a1, a2;

return 0;

}

a) Run-time Error  
b) Runs perfectly  
c) Segmentation fault  
d) Compile-time error

Answer: d  
Explanation: .(dot) operator cannot be overloaded therefore the program gives error.

# C++ Programming Questions and Answers – Operator Overloading – 2

1. What is a binary operator?  
a) Operator that performs its action on a single operand  
b) Operator that performs its action on two operand  
c) Operator that performs its action on three operand  
d) Operator that performs its action on any number of operands

Answer: b  
Explanation: As the word binary itself means 2 therefore a binary operator operates on two operands.

2. Which is the correct example of a binary operator?  
a) ++  
b) —  
c) Dereferencing operator(\*)  
d) +

Answer: d  
Explanation: +(adding two operands) requires two operands whereas ++(increases value by 1), –(decreases value by 1) and \*(dereferencing operator used for accessing value of pointers) requires only one operand.

3. Which is the correct example of a unary operator?  
a) &  
b) ==  
c) —  
d) /

Answer: c  
Explanation: &, == and / requires two operands whereas — requires only one operand, in general, it decreases the value of operand by 1.

4. Which is called ternary operator?  
a) ?:  
b) &&  
c) |||  
d) ===

Answer: a  
Explanation: ?: is called ternary operator because it separates three expressions. exp1 ? exp2 : exp3.

5. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class complex

{

int i;

int j;

public:

complex(int a, int b)

{

i = a;

j = b;

}

complex operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

return temp;

}

void show(){

cout<<"Complex Number: "<<i<<" + i"<<j<<endl;

}

};

int main(int argc, char const \*argv[])

{

complex c1(1,2);

complex c2(3,4);

complex c3 = c1 + c2;

c3.show();

return 0;

}

a) 4 + i6  
b) 2 + i2  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: In the operator overloaded function we are trying to call default constructor of the class complex but as we have overridden the constructor by our constructor therefore the default constructor cannot be called hence the program gives error.

6. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class complex

{

int i;

int j;

public:

complex(){}

complex(int a, int b)

{

i = a;

j = b;

}

complex operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

return temp;

}

void show(){

cout<<"Complex Number: "<<i<<" + i"<<j<<endl;

}

};

int main(int argc, char const \*argv[])

{

complex c1(1,2);

complex c2(3,4);

complex c3 = c1 + c2;

c3.show();

return 0;

}

a) Complex Number: 4 + i6  
b) Complex Number: 2 + i2  
c) Error  
d) Segmentation fault

Answer: a  
Explanation: As we have defined in the class complec that when we add the two objects of the class complex then add those two complex numbers and show() displays that result.

7. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class complex

{

int i;

int j;

public:

complex(){}

complex(int a, int b)

{

i = a;

j = b;

}

complex operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

return temp;

}

void operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

temp.show\_poss();

}

void show(){

cout<<"Complex Number: "<<i<<" + i"<<j<<endl;

}

void show\_poss(){

cout<<"Your result after addition will be: "<<i<<" + i"<<j<<endl;

}

};

int main(int argc, char const \*argv[])

{

complex c1(1,2);

complex c2(3,4);

c1 + c2;

return 0;

}

a) Complex Number: 4 + i6  
b) Complex Number: 2 + i2  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: Each operator function can be defined only once in a class. So as in this program we are trying to define two functions for operator ‘+’ which is not allowed in C++ therefore program gives error.

8. Given the following C++ code. How would you define the < operator for Box class so that when boxes b1 and b2 are compared in if block the program gives correct result?

#include <iostream>

#include <string>

using namespace std;

class Box

{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 < b2){

cout<<"Box 2 has large capacity.";

}

else{

cout<<"Box 1 has large capacity.";

}

return 0;

}

a)

bool operator<(Box b)

{

return this->capacity < b.capacity ? true : false;

}

b)

bool operator<(Box b)

{

return this->capacity > b.capacity ? true : false;

}

c)

bool operator<(Box b)

{

return b1 > b2 ? true : false;

}

d)

bool operator<(Box b)

{

return this < b ? true : false;

}

Answer: a  
Explanation: As we need to give the result after comparing the capacity of two boxes. We use < operator and as this is the first operand and second operand is passed so we need to do this->capacity < b.capacity (passed object) to make the program run.

9. Which is the correct statement about operator overloading?  
a) Only arithmetic operators can be overloaded  
b) Only non-arithmetic operators can be overloaded  
c) Precedence of operators are changed after overlaoding  
d) Associativity and precedence of operators does not change

Answer: d  
Explanation: Both arithmetic and non-arithmetic operators can be overloaded. The precedence and associativity of operators remains the same after and before operator overloading.

10. Pick the incorrect statements out of the following.  
a) Operator overloading does not disturbs the precedence of operators  
b) Arity of operators can be changed using operator overloading  
c) No new operators can be created  
d) All of the mentioned

Answer: b  
Explanation: Arity means a number of operands an operator requires to perform its action and operator overloading does not changes the arity of any operator.

11. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Box

{

int capacity;

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

return 0;

}

a) Error  
b) Segmentation fault  
c) 4  
d) No output

Answer: a  
Explanation: As constructors are defined private and we know objects cannot access private objects therefore program gives error. Also no class should have private constructor.

12. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

bool operator<(Box b){

return this->capacity < b.capacity ? true : false;

}

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 < b2){

cout<<"Box 2 has large capacity.";

}

else{

cout<<"Box 1 has large capacity.";

}

return 0;

}

a) Error  
b) Segmentation fault  
c) Box 2 has large capacity  
d) No output

Answer: a  
Explanation: As the operator overloaded function defined is private therfore on comparison the function cannot be called from outside therefore the program gives error.

13. Which operator should be overloaded in the following code to make the program error free?

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 == b2){

cout<<"Equal";

}

else{

cout<<"Not Equal";

}

return 0;

}

a) +  
b) ==  
c) =  
d) ()

Answer: b  
Explanation: As in the if block we are trying to compare two Box objects and no method is defined to tell compiler how the comparison should be done bwteen these two objects. Hence we need to overload the == operator.

14. Give the function prototype of the operator function which we need to define in this program so that the program has no errors.

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 == b2){

cout<<"Equal";

}

else{

cout<<"Not Equal";

}

return 0;

}

a) bool operator==();  
b) bool operator==(Box b){}  
c) bool operator==(Box b);  
d) Box operator==();

Answer: c  
Explanation: In this question we are asked to give the function prototypr not the function definition so the answer should not contain {} braces. The correct overloaded function is bool operator==(Box b);

15. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

bool operator<(Box b){

return b.capacity < this->capacity? true : false;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 < b2){

cout<<"B1's capacity is small";

}

else{

cout<<"B2's capacity is small";

}

return 0;

}

1. B1's capacity is small  
   b) B2's capacity is small  
   c) Error  
   d) Segmentation fault  
   Answer: b  
   Explanation: Though the b1's capacity is small the program prints B2's capacity is small because in the < operator overloaded function we are checking B2's capacity < B1's capacity which is false therefore the else is executed.

**C++ Programming Questions and Answers – Complex Number Type**

1. Which header file is used to declare the complex number?  
a) complexnum  
b) complex  
c) complex number  
d) complexarg

Answer: b  
Explanation: <complex> header file is used for declaring a complex number in C++.

2. How to declare the complex number?  
a) (3, 4)  
b) complex(3, 4)  
c) (3, 4i)  
d) (3, 4g)

Answer: b  
Explanation: We can declare the complex number by using complex(3,4) where 3 is a real number and 4 is imaginary part.

3. How many real types are there in complex numbers?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three real types in complex numbers. They are float complex, double complex, long double complex.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <complex>
3. using namespace std;
4. int main()
5. {
6. complex<double> c1(4.0, 16.0), c2;
7. c2 = pow(c1, 2.0);
8. cout << c2;
9. return 0;
10. }

a) (-240, 128)  
b) (240, 128)  
c) (240, 120)  
d) (240, -122)

Answer: a  
Explanation: In this program, we are finding the square of the complex number.  
Output:

$ g++ comp.cpp

$ a.out

(-240,128)

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <complex>
3. using namespace std;
4. int main()
5. {
6. complex<double> c\_double(2, 3);
7. complex<int> c\_int(4, 5);
8. c\_double \*= 2;
9. c\_double = c\_int;
10. cout << c\_double;
11. return 0;
12. }

a) (2, 3)  
b) (4, 5)  
c) (8, 15)  
d) (8, 10)

Answer: b  
Explanation: We are just copying the value of c\_int into c\_double, So it’s printing as (4,5).  
Output:

$ g++ comp1.cpp

$ a.out

(4,5)

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <complex>
3. using namespace std;
4. int main()
5. {
6. complex<int> i(2, 3);
7. i = i \* 6 / 3;
8. cout << i;
9. return 0;
10. }

a) (4, 6)  
b) (2, 3)  
c) (6, 12)  
d) (6, 15)

Answer: a  
Explanation: We are multiplying the complex number by 2.  
Output:

$ g++ comp2.cpp

$ a.out

(4,6)

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <complex>
3. using namespace std;
4. int main()
5. {
6. complex<double> c1(4.0,3.0);
7. cout << "c1: " << c1;
8. complex<float> c2(polar(5.0,0.75));
9. cout << c1 + complex<double>(c2.real(),c2.imag());
10. return 0;
11. }

a) c1: (4,3)(7.65844,6.40819)  
b) c1: (4,3)(7,6)  
c) both c1: (4,3)(7.65844,6.40819) & c1: (4,3)(7,6)  
d) c1: (5,3)(7,6)

Answer: a  
Explanation: We are adding the two complex numbers and printing the result.  
Output:

$ g++ comp3.cpp

$ a.out

c1: (4,3)(7.65844,6.40819)

8. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <complex>
3. using namespace std;
4. int main()
5. {
6. complex<double> c1(4.0, 3.0);
7. complex<float> c2(polar(5.0, 0.75));
8. cout << (c1 += sqrt(c1)) << endl;
9. return 0;
10. }

a) (4.0, 3.0)  
b) (6.12132, 3.70711)  
c) (5.0, 0.75)  
d) (5.0, 3.75)  
View Answer

Answer: b  
Explanation: In this program, we are adding both complex number and finding the square root of it.  
Output:

$ g++ comp4.cpp

$ a.out

(6.12132,3.70711)

9. Which of the following is not a function of complex values?  
a) real  
b) imag  
c) norm  
d) cartesian

Answer: d  
Explanation: Real is used for returning real part, imag for imaginary part and norm for calculating norm of a complex number. There is no such function Cartesian in complex header file.

10. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <complex>
3. using namespace std;
4. int main ()
5. {
6. complex<double> mycomplex (20.0, 2.0);
7. cout << imag(mycomplex) << endl;
8. return 0;
9. }

a) 2  
b) 20  
c) 40  
d) 30

Answer: a  
Explanation: imag part will return the imaginary part of the complex number.  
Output:

$ g++ comp5.cpp

$ a.out

2

**C++ Programming Questions and Answers – Conversion Operators**

1. What is the return type of the conversion operator?  
a) void  
b) int  
c) float  
d) no return type

Answer: d  
Explanation: Conversion operator doesn’t have any return type not even void.

2. Why we use the “dynamic\_cast” type conversion?  
a) result of the type conversion is a valid  
b) to be used in low memory  
c) result of the type conversion is an invalid  
d) it is used for storage

Answer: a  
Explanation: It is used to check that operators and operands are compatible after conversion.

3. How many parameters does a conversion operator may take?  
a) 0  
b) 1  
c) 2  
d) as many as possible

Answer: a  
Explanation: 0 parameters does a conversion operator will take.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample1
4. {
5. float i, j;
6. };
7. class sample2
8. {
9. int x, y;
10. public:
11. sample2 (int a, int b)
12. {
13. x = a;
14. y = b;
15. }
16. int result()
17. {
18. return x + y;
19. }
20. };
21. int main ()
22. {
23. sample1 d;
24. sample2 \* padd;
25. padd = (sample2\*) &d;
26. cout<< padd->result();
27. return 0;
28. }

a) 20  
b) runtime error  
c) random number  
d) runtime error or random number

Answer: d  
Explanation: As it assigns to a reference to an object of another incompatible type using explicit type-casting.  
Output:

$ g++ con.cpp

$ a.out

14032334

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample
4. {
5. public:
6. sample(int i) : m\_i(i) { }
7. public:
8. int operator()(int i = 0) const
9. {
10. return m\_i + i;
11. }
12. operator int () const
13. {
14. return m\_i;
15. }
16. private:
17. int m\_i;
18. friend int g(const sample&);
19. };
20. int f(char c)
21. {
22. return c;
23. }
24. int main()
25. {
26. sample f(2);
27. cout << f(2);
28. return 0;
29. }

a) 3  
b) 4  
c) 5  
d) 6

Answer: b  
Explanation: In this program, we are adding its value with it itself, So only we got the output as 4.  
Output:

$ g++ con1.cpp

$ a.out 4

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <cmath>
3. using namespace std;
4. class Complex
5. {
6. private:
7. double real;
8. double imag;
9. public:
10. Complex(double r = 0.0, double i = 0.0) : real(r), imag(i)
11. {}
12. double mag()
13. {
14. return getMag();
15. }
16. operator double ()
17. {
18. return getMag();
19. }
20. private:
21. double getMag()
22. {
23. return sqrt(real \* real + imag \* imag);
24. }
25. };
26. int main()
27. {
28. Complex com(3.0, 4.0);
29. cout << com.mag();
30. cout << com;
31. return 0
32. }

a) 5 5  
b) 4 5  
c) 6 6  
d) 7 5

Answer: a  
Explanation: In this program, we are calculating the magnitude value by two ways.  
Output:

$ g++ con3.cpp

$ a.out

55

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. class test
5. {
6. public:
7. operator string ()
8. {
9. return "Converted";
10. }
11. };
12. int main()
13. {
14. test t;
15. string s = t;
16. cout << s << endl;
17. return 0;
18. }

a) converted  
b) error  
c) run time error  
d) convertedconverted

Answer: a  
Explanation: In this program, We casted the string to the object of the class.  
Output:

$ g++ con4.cpp

$ a.out

converted

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. double a = 21.09399;
6. float b = 10.20;
7. int c ;
8. c = (int) a;
9. cout << c ;
10. c = (int) b;
11. cout << c ;
12. return 0;
13. }

a) 2110  
b) 1210  
c) 21  
d) 121

Answer: a  
Explanation: In this program, we casted the data type to integer.  
Output:

$ g++ con5.cpp

$ a.out

2110

9. How are types therein user-defined conversion?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: There are two types of user-defined conversions. They are conversion by the constructor, Conversion functions.

10. Pick out the correct syntax of operator conversion.  
a) operator float()const  
b) operator float()  
c) operator const  
d) operator const()

Answer: a  
Explanation: The syntax of operator conversion is operator float()const.

**C++ Programming Questions and Answers – Friends**

1. Which rule will not affect the friend function?  
a) private and protected members of a class cannot be accessed from outside  
b) private and protected member can be accessed anywhere  
c) protected member can be accessed anywhere  
d) private member can be accessed anywhere

Answer: a  
Explanation: Friend is used to access private and protected members of a class from outside the same class.

2. Which keyword is used to declare the friend function?  
a) firend  
b) friend  
c) classfriend  
d) myfriend

Answer: b  
Explanation: friend keyword is used to declare a friend function in C++.

3. What is the syntax of friend function?  
a) friend class1 Class2;  
b) friend class;  
c) friend class  
d) friend class()

Answer: a  
Explanation: In option a, the class2 is the friend of class1 and it can access all the private and protected members of class1.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Box
4. {
5. double width;
6. public:
7. friend void printWidth( Box box );
8. void setWidth( double wid );
9. };
10. void Box::setWidth( double wid )
11. {
12. width = wid;
13. }
14. void printWidth( Box box )
15. {
16. box.width = box.width \* 2;
17. cout << "Width of box : " << box.width << endl;
18. }
19. int main( )
20. {
21. Box box;
22. box.setWidth(10.0);
23. printWidth( box );
24. return 0;
25. }

a) 40  
b) 5  
c) 10  
d) 20

Answer: d  
Explanation: We are using the friend function for printwidth and multiplied the width value by 2, So we got the output as 20  
Output:

$ g++ friend.cpp

$ a.out

20

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample
4. {
5. int width, height;
6. public:
7. void set\_values (int, int);
8. int area () {return (width \* height);}
9. friend sample duplicate (sample);
10. };
11. void sample::set\_values (int a, int b)
12. {
13. width = a;
14. height = b;
15. }
16. sample duplicate (sample rectparam)
17. {
18. sample rectres;
19. rectres.width = rectparam.width \* 2;
20. rectres.height = rectparam.height \* 2;
21. return (rectres);
22. }
23. int main ()
24. {
25. sample rect, rectb;
26. rect.set\_values (2, 3);
27. rectb = duplicate (rect);
28. cout << rectb.area();
29. return 0;
30. }

a) 20  
b) 16  
c) 24  
d) 18

Answer: c  
Explanation: In this program, we are using the friend function for duplicate function and calculating the area of the rectangle.  
Output:

$ g++ friend1.cpp

$ a.out

24

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample;
4. class sample1
5. {
6. int width, height;
7. public:
8. int area ()
9. {
10. return (width \* height);}
11. void convert (sample a);
12. };
13. class sample
14. {
15. private:
16. int side;
17. public:
18. void set\_side (int a)
19. {
20. side = a;
21. }
22. friend class sample1;
23. };
24. void sample1::convert (sample a)
25. {
26. width = a.side;
27. height = a.side;
28. }
29. int main ()
30. {
31. sample sqr;
32. sample1 rect;
33. sqr.set\_side(6);
34. rect.convert(sqr);
35. cout << rect.area();
36. return 0;
37. }

a) 24  
b) 35  
c) 16  
d) 36

Answer: d  
Explanation: In this program, we are using the friend for the class and calculating the area of the square.  
Output:

$ g++ friend2.cpp

$ a.out

36

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class base
4. {
5. int val1, val2;
6. public:
7. int get()
8. {
9. val1 = 100;
10. val2 = 300;
11. }
12. friend float mean(base ob);
13. };
14. float mean(base ob)
15. {
16. return float(ob.val1 + ob.val2) / 2;
17. }
18. int main()
19. {
20. base obj;
21. obj.get();
22. cout << mean(obj);
23. return 0;
24. }

a) 200  
b) 150  
c) 100  
d) 300

Answer: a  
Explanation: In this program, We are finding the mean value by declaring the function mean as a friend of class base.  
Output:

$ g++ friend3.cpp

$ a.out

200

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class sample
4. {
5. private:
6. int a, b;
7. public:
8. void test()
9. {
10. a = 100;
11. b = 200;
12. }
13. friend int compute(sample e1);
14. };
15. int compute(sample e1)
16. {
17. return int(e1.a + e1.b) - 5;
18. }
19. int main()
20. {
21. sample e;
22. e.test();
23. cout << compute(e);
24. return 0;
25. }

a) 100  
b) 200  
c) 300  
d) 295

Answer: d  
Explanation: In this program, we are finding a value from the given function by using the friend for compute function.  
Output:

$ g++ friend4.cpp

$ a.out

295

9. Pick out the correct statement.  
a) A friend function may be a member of another class  
b) A friend function may not be a member of another class  
c) A friend function may or may not be a member of another class  
d) None of the mentioned

Answer: c  
Explanation: A friend function may or may not be a member of another class is the correct statement.

10. Where does keyword ‘friend’ should be placed?  
a) function declaration  
b) function definition  
c) main function  
d) block function

Answer: a  
Explanation: The keyword friend is placed only in the function declaration of the friend function and not in the function definition because it is used toaccess the member of a class.

**C++ Programming Questions and Answers – Friend Function**

1. What is a friend function in C++?  
a) A function which can access all the private, protected and public members of a class  
b) A function which is not allowed to access any member of any class  
c) A function which is allowed to access public and protected members of a class  
d) A function which is allowed to access only public members of a class

Answer: a  
Explanation: Friend function in C++ is a function which can access all the private, protected and public members of a class.

2. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Box

{

int capacity;

public:

Box(int cap){

capacity = cap;

}

friend void show();

};

void show()

{

Box b(10);

cout<<"Value of capacity is: "<<b.capacity<<endl;

}

int main(int argc, char const \*argv[])

{

show();

return 0;

}

a) Value of capacity is: 10  
b) Value of capacity is: 100  
c) Error  
d) Segmentation fault

Answer: a  
Explanation: As show() is a friend function of class Box hence any object from this function can access the private member of the class Box.

3. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Box

{

int capacity;

public:

Box(int cap){

capacity = cap;

}

friend void show();

};

void Box::show()

{

Box b(10);

cout<<"Value of capacity is: "<<b.capacity<<endl;

}

int main(int argc, char const \*argv[])

{

show();

return 0;

}

a) Value of capacity is: 10  
b) Value of capacity is: 100  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: Though it is used to declare the friend functions inside classes they are not members of any class therefore when we giving the definition to friend function show() we should not use Box::show() way of defining it.

4. How many member functions are there in this C++ class excluding constructors and destructors?

class Box

{

int capacity;

public:

void print();

friend void show();

bool compare();

friend bool lost();

};

a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: A friend functions are not members of any class. Hence this class has only 2 member functions.

5. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class B

{

int b;

public:

B(int i){

b = i;

}

};

class C

{

B b;

public:

C(int i){

b = B(i);

}

friend void show();

};

void show()

{

C c(10);

cout<<"value of b is: "<<c.b.b<<endl;

}

int main(int argc, char const \*argv[])

{

show();

return 0;

}

a) value of b is: 10  
b) value of b is: 12345435  
c) error  
d) segmentation fault

Answer: c  
Explanation: There is two error in the program. First the program doesn’t have a default constructor for the class B which is used when the object of B is declared inside the class C. Second show() is friend function of class C therefore it can access only private member of class C, not B therefore when we are doing c.b.b here the last b is private member of class B which is not accessible.

6. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class B

{

int b;

public:

B(){}

B(int i){

b = i;

}

int show(){

return b;

}

};

class C

{

B b;

public:

C(int i){

b = B(i);

}

friend void show();

};

void show()

{

C c(10);

cout<<"value of b is: "<<c.b.show()<<endl;

}

int main(int argc, char const \*argv[])

{

show();

return 0;

}

a) value of b is: 10  
b) value of b is: 12345435  
c) error  
d) segmentation fault

Answer: a  
Explanation: The program follows correct syntax and semantics hence no errors.

7. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class B

{

int b;

public:

B(){}

B(int i){

b = i;

}

int show(){

return b;

}

};

class C

{

B b;

public:

C(int i){

b = B(i);

}

friend void show(){

C c(10);

cout<<"value of b is: "<<c.b.show()<<endl;

}

};

int main(int argc, char const \*argv[])

{

show();

return 0;

}

a) value of b is: 10  
b) value of b is: 12345435  
c) error  
d) segmentation fault

Answer: c  
Explanation: No function show() is defined in the scope of main() function.

8. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class B

{

int b;

public:

B(){}

B(int i){

b = i;

}

int show(){

return b;

}

};

class C

{

B b;

public:

C(int i){

b = B(i);

}

friend void show(){

C c(10);

cout<<"value of b is: "<<c.b.show()<<endl;

}

};

int main(int argc, char const \*argv[])

{

C c(1);

c.show();

return 0;

}

a) value of b is: 10  
b) value of b is: 12345435  
c) error  
d) segmentation fault

Answer: c  
Explanation: Friend functions are not members of any class therefore they should not be called using class objects.

9. Pick the correct statement.  
a) Friend functions are in the scope of a class  
b) Friend functions can be called using class objects  
c) Friend functions can be invoked as a normal function  
d) Friend functions can access only protected members not the private members

Answer: c  
Explanation: Friend functions are not in the scope of a class and hence cannot be called through a class object. A friend function can access all types of members of the class. They can be invoked as a normal function.

10. Which of the following is correct about friend functions?  
a) Friend functions use the dot operator to access members of a class using class objects  
b) Friend functions can be private or public  
c) Friend cannot access the members of the class directly  
d) All of the mentioned

Answer: a  
Explanation: Friends function are defined outside the class, hence there is no concept of private or public for them. Since they are friend of a class, they can access private or protected members of the class directly using the dot operator.

11. Which keyword is used to represent a friend function?  
a) friend  
b) Friend  
c) friend\_func  
d) Friend\_func

Answer: a  
Explanation: friend keyword is used to declare a friend function.