**Quiz 5**

1. **How can we restrict dynamic allocation of objects of a class using new?**

|  |  |
| --- | --- |
| **A** | By overloading new operator |
| **B** | By making an empty private new operator |
| **C** | By making an empty private new and new[] operators |
| **D** | By overloading new operator and new[] operators |

1. **Which of the following operators cannot be overloaded**

|  |  |
| --- | --- |
| **A** | . (Member Access or Dot operator) |
| **B** | ?: (Ternary or Conditional Operator ) |
| **C** | :: (Scope Resolution Operator) |
| **D** | .\* (Pointer-to-member Operator ) |
| **E** | All of the above |

1. **Which of the following operators are overloaded by default by the compiler?**

1) Comparison Operator ( == )

2) Assignment Operator ( = )

|  |  |
| --- | --- |
| **A** | Both 1 and 2 |
| **B** | Only 1 |
| **C** | Only 2 |
| **D** | None of the two |

1. **Which of the following operators should be preferred to overload as a global function rather than a member method?**

|  |  |
| --- | --- |
| **A** | Postfix ++ |
| **B** | Comparison Operator |
| **C** | Insertion Operator << |
| **D** | Prefix++ |

1. **How does C++ compiler differs between overloaded postfix and prefix operators?**

|  |  |
| --- | --- |
| **A** | C++ doesn't allow both operators to be overlaoded in a class |
| **B** | A postfix ++ has a dummy parameter |
| **C** | A prefix ++ has a dummy parameter |
| **D** | By making prefix ++ as a global function and postfix as a member function. |

1. **Predict the output**

*#include<iostream>*

*using namespace std;*

*class A*

*{*

*int i;*

*public:*

*A(int ii = 0) : i(ii) {}*

*void show() {  cout << i << endl;  }*

*};*

*class B*

*{*

*int x;*

*public:*

*B(int xx) : x(xx) {}*

*operator A() const {  return A(x); }*

*};*

*void g(A a)*

*{*

*a.show();*

*}*

*int main()*

*{*

*B b(10);*

*g(b);*

*g(20);*

*return 0;*

*}*

**Output:**

|  |  |
| --- | --- |
| **A** | Compiler Error |
| **B** | 10  20 |
| **C** | 20  20 |
| **D** | 10  10 |

1. **Output of following program?**

*#include <iostream>*

*using namespace std;*

*class Test2*

*{*

*int y;*

*};*

*class Test*

*{*

*int x;*

*Test2 t2;*

*public:*

*operator Test2 ()  { return t2; }*

*operator int () { return x; }*

*};*

*void fun ( int x) { cout << "fun(int) called"; }*

*void fun ( Test2 t ) { cout << "fun(Test 2) called"; }*

*int main()*

*{*

*Test t;*

*fun(t);*

*return 0;*

*}*

|  |  |
| --- | --- |
| **A** | fun(int) called |
| **B** | fun(Test 2) called |
| **C** | Compiler Error: Ambiguous call to fun() |

1. **Predict the output?**

|  |
| --- |
| *#include<stdlib.h>*  *#include<stdio.h>*  *#include<iostream>*    *using namespace std;*    *class Test {*  *int x;*  *public:*  *void\* operator new(size\_t size);*  *void operator delete(void\*);*  *Test(int i) {*  *x = i;*  *cout << "Constructor called \n";*  *}*  *~Test() { cout << "Destructor called \n"; }*  *};*      *void\* Test::operator new(size\_t size)*  *{*  *void \*storage = malloc(size);*  *cout << "new called \n";*  *return storage;*  *}*    *void Test::operator delete(void \*p )*  *{*  *cout<<"delete called \n";*  *free(p);*  *}*    *int main()*  *{*  *Test \*m = new Test(5);*  *delete m;*  *return 0;*  *}* |

**Output:**

|  |  |
| --- | --- |
| **A** | new called  Constructor called  delete called  Destructor called |
| **B** | new called  Constructor called  Destructor called  delete called |
| **C** | Constructor called  new called  Destructor called  delete called |
| **D** | Constructor called  new called  delete called  Destructor called |



|  |  |
| --- | --- |
|  | *#include<iostream>*  *using namespace std;*    *class Point {*  *private:*  *int x, y;*  *public:*  *Point() : x(0), y(0) { }*  *Point& operator()(int dx, int dy);*  *void show() {cout << "x = " << x << ", y = " << y; }*  *};*    *Point& Point::operator()(int dx, int dy)*  *{*  *x = dx;*  *y = dy;*  *return \*this;*  *}*    *int main()*  *{*  *Point pt;*  *pt(3, 2);*  *pt.show();*  *return 0;*  *}* |

**Output:**

|  |  |
| --- | --- |
| **A** | x = 3, y = 2 |
| **B** | Compiler Error |
| **C** | x = 2, y = 3 |

1. **Which of the following operator functions cannot be global, i.e., must be a member function.**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **A** | new |
| **B** | delete |
| **C** | Converstion Operator |
| **D** | All of the above |

**Result:**

1. C

2. E.

3. C

4. C cout is an object of ostream class which is a compiler defined class. When we do "cout << obj" where obj is an object of our class, the compiler first looks for an operator function in ostream, then it looks for a global function. One way to overload insertion operator is to modify ostream class which may not be a good idea. So we make a global method.

5. B

6. B Note that the class B has as conversion operator overloaded, so an object of B can be converted to that of A. Also, class A has a constructor which can be called with single integer argument, so an int can be converted to A.

7. C

8. B Consider the following statement

*Test \*ptr = new Test;*

There are actually two things that happen in the above statement--memory allocation and object construction; **the new keyword** is responsible for both. One step in the process is to call **operator new** in order to allocate memory; the other step is to actually invoke the constructor. **Operator new** only allows us to change the memory allocation method, but does not do anything with the constructor calling method. **Keyword new** is responsible for calling the constructor, not **operator new**.

9. A This a simple example of function call operator overloading. The function call operator, when overloaded, does not modify how functions are called. Rather, it modifies how the operator is to be interpreted when applied to objects of a given type. If you overload a function call operator for a class its declaration will have the following form:

*return\_type operator()(parameter\_list)*

10. C.