Can a destructor be virtual? Will the following program compile?

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;  class Base {  public:    virtual ~Base() {}  };  int main() {     return 0;  } | |
| A | Yes |
| B | No |

Explanation:

A destructor can be virtual. We may want to call appropriate destructor when a base class pointer points to a derived class object and we delete the object. If destructor is not virtual, then only the base class destructor may be called. For example, consider the following program.

// Not good code as destructor is not virtual

#include<iostream>

using namespace std;

class Base {

public:

Base() { cout << "Constructor: Base" << endl; }

~Base() { cout << "Destructor : Base" << endl; }

};

class Derived: public Base {

public:

Derived() { cout << "Constructor: Derived" << endl; }

~Derived() { cout << "Destructor : Derived" << endl; }

};

int main() {

Base \*Var = new Derived();

delete Var;

return 0;

}

Output on GCC:

Constructor: Base

Constructor: Derived

Destructor : Base

|  |
| --- |
| Question 10 |

|  |  |
| --- | --- |
| #include<iostream>  using namespace std;  class Base  {  public:      Base()    { cout<<"Constructor: Base"<<endl; }      virtual ~Base()   { cout<<"Destructor : Base"<<endl; }  };  class Derived: public Base {  public:      Derived()   { cout<<"Constructor: Derived"<<endl; }      ~Derived()  { cout<<"Destructor : Derived"<<endl; }  };  int main()  {      Base \*Var = new Derived();      delete Var;      return 0;  } | |
| A | Constructor: Base  Constructor: Derived  Destructor : Derived  Destructor : Base |
| B | Constructor: Base  Constructor: Derived  Destructor : Base |
| C | Constructor: Base  Constructor: Derived  Destructor : Derived |
| D | Constructor: Derived  Destructor : Derived |

Explanation:

Since the destructor is vitrual, the derived class destructor is called which in turn calls base class destructor.