|  |
| --- |
| Question 1 |

Predict the output of following program.

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;  class A  {  protected:      int x;  public:      A() {x = 0;}      friend void show();  };    class B: public A  {  public:      B() : y (0) {}  private:      int y;  };    void show()  {      A a;      B b;      cout << "The default value of A::x = " << a.x << " ";      cout << "The default value of B::y = " << b.y;  } | |
| A | Compiler Error in show() because x is protected in class A |
| B | Compiler Error in show() because y is private in class b |
| C | The default value of A::x = 0 The default value of B::y = 0 |
| D | Compiler Dependent |

Explanation:

Please note that show() is a friend of class A, so there should not be any compiler error in accessing any member of A in show(). Class B is inherited from A, the important point to note here is friendship is not inherited. So show() doesn't become a friend of B and therefore can't access private members of B.

|  |
| --- |
| Question 2 |

Predict the output the of following program.

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;    class B;  class A {      int a;  public:      A():a(0) { }      void show(A& x, B& y);  };    class B {  private:      int b;  public:      B():b(0) { }      friend void A::show(A& x, B& y);  };    void A::show(A& x, B& y) {      x.a = 10;      cout << "A::a=" << x.a << " B::b=" << y.b;  }    int main() {      A a;      B b;      a.show(a,b);      return 0;  } | |
| A | Compiler Error |
| B | A::a=10 B::b=0 |
| C | A::a=0 B::b=0 |

Explanation:

This is simple program where a function of class A is declared as friend of class B. Since show() is friend, it can access private data members of B.

**Friend class and function in C++**

**Friend Class** A friend class can access private and protected members of other class in which it is declared as friend. It is sometimes useful to allow a particular class to access private members of other class. For example a LinkedList class may be allowed to access private members of Node.

|  |
| --- |
| class Node  {  private:    int key;    Node \*next;    /\* Other members of Node Class \*/    **friend class LinkedList;** // Now class  LinkedList can                             // access private members of Node  }; |

**Friend Function** Like friend class, a friend function can be given special grant to access private and protected members. A friend function can be:  
a) A method of another class  
b) A global function

|  |
| --- |
| class Node  {  private:    int key;    Node \*next;      /\* Other members of Node Class \*/    friend int LinkedList::search(); // Only search() of linkedList                                    // can access internal members  }; |

**Following are some important points about friend functions and classes:**  
**1)** Friends should be used only for limited purpose. too many functions or external classes are declared as friends of a class with protected or private data, it lessens the value of encapsulation of separate classes in object-oriented programming.

**2)** Friendship is not mutual. If a class A is friend of B, then B doesn’t become friend of A automatically.

**3)** Friendship is not inherited (See [this](https://www.geeksforgeeks.org/g-fact-34/) for more details)

**4)** The concept of friends is not there in Java.

**A simple and complete C++ program to demonstrate friend Class**

|  |
| --- |
| #include <iostream>  class A {  private:      int a;  public:      A() { a=0; }      friend class B;     // Friend Class  };    class B {  private:      int b;  public:      void showA(A& x) {          // Since B is friend of A, it can access          // private members of A          std::cout << "A::a=" << x.a;      }  };    int main() {     A a;     B b;     b.showA(a);     return 0;  } |

Output:

A::a=0

**A simple and complete C++ program to demonstrate friend function of another class**

|  |
| --- |
| #include <iostream>    class B;    class A  {  public:      void showB(B& );  };    class B  {  private:      int b;  public:      B()  {  b = 0; }      friend void A::showB(B& x); // Friend function  };    void A::showB(B &x)  {      // Since show() is friend of B, it can      // access private members of B      std::cout << "B::b = " << x.b;  }    int main()  {      A a;      B x;      a.showB(x);      return 0;  } |

Output:

B::b = 0

**A simple and complete C++ program to demonstrate global friend**

|  |
| --- |
| #include <iostream>    class A  {      int a;  public:      A() {a = 0;}      friend void showA(A&); // global friend function  };    void showA(A& x) {      // Since showA() is a friend, it can access      // private members of A      std::cout << "A::a=" << x.a;  }    int main()  {      A a;      showA(a);      return 0;  } |

Output:

A::a = 0