Friend function in C++

For the sake of data-hiding, we emphasize the use of a private modifier for critical data-members. However, there might be some situations where multiple classes need to work together closely, so much so that they require access to each other's private members too. Well, we can solve the problem by using getters and setters, but it would be tedious to do so for all the private members. Instead, we use the ***friend*** keyword to provide access to the private fields to outside entities (global functions, other class-member functions etc.).

Friend Class

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

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
    private:
        int key;
        Node *next;
        Node(int key) : key(key), next(nullptr) {}

    public:
        friend class LinkedList;
};

class LinkedList
{
    public:
        Node *root;

        LinkedList(int key) {
            root = new Node(key);
```

```
        }

        void insert(int key) {
            Node *trav = root;
            while (trav->next != nullptr)
                trav = trav->next;
            trav->next = new Node(key);
        }

        void print() {
            Node *trav = root;
            while (trav != nullptr) {
                cout << trav->key << " ";
                trav = trav->next;
            }
            cout << endl;
        }
};

int main()
{
     LinkedList list(0);

     list.insert(1);
     list.insert(2);
     list.insert(3);
     list.insert(4);

     list.print();

     return 0;
}
```

Output:

```
0 1 2 3 4
```

As we can see in the above code, *LinkedList* has access to all the private fields of *Node* class, because it has been declared as a friend inside the Node class.

Friend Function

Like a friend class, a friend function can be given special access to private and protected members. A friend function can be:

- A Member Function of another class

```
#include <bits/stdc++.h>
using namespace std;

//forward-declaration is
//necessary for usage in A
//as B is not defined yet
class B;

class A
{
    public:
        void showB(B &x);
};

class B
{
    private:
        int b;
    public:
        B() : b(0) {}

        //Friend function Declaration
        friend void A::showB(B &x);
};

//Friend Member Function Definition
void A::showB(B &x)
{
    cout << "B::b = " << x.b;
}
```

```
int main()
{
    A a;
    B x;

    a.showB(x);

    return 0;
}
```

Output:

```
B::b = 0
```

- A global Function

```
#include <bits/stdc++.h>
using namespace std;

class A
{
    private:
        int a;
    public:
        A() : a(0) {}

        //global friend function
        friend void showA(A&);
};

void showA(A &x) {
    std::cout << "A::a=" << x.a;
}

int main()
{
    A a;
```

```
    showA(a);
    return 0;
}
```

Output:

`A::a=0`