# /*Iterator*/

```
#include <iostream>
using namespace std;

class List
{
	struct Node
	{
		int value;
		Node *next;
		Node(int value):value(value),next(nullptr)
		{
		}
	};
public:
	Node *head;
	Node *tail;

	List():head(nullptr),tail(nullptr)
	{
	}

	class Iterator
	{
	private:
		Node *current;
	public :
		Iterator(List *ptr)
		{
			if(nullptr==ptr)
				current=nullptr;
			else
				current=ptr->head;
		}

		Iterator operator++(int)
		{
			current=current->next;
			return *this;
		}

		int operator*()
		{
			return current->value;
		}

		bool operator==(const Iterator &ptr)
		{
			return this->current==ptr.current;
		}
		bool operator!=(const Iterator &ptr)
		{
			return this->current!=ptr.current;
		}
	};

	Iterator begin()
	{
		return Iterator(this);
	}
```

```
        Iterator end()
        {
                return Iterator(nullptr);
        }

        void push_Back(int value)
        {
                Node *newNode = new Node(value);

                if (nullptr == tail)
                        head = newNode;
                else
                        tail->next = newNode;
                tail = newNode;
        }

        void push_Front(int value)
        {
                Node *newNode = new Node(value);

                if (nullptr == head)
                        tail = newNode;
                else
                        newNode->next = head;
                head = newNode;
        }
};

int main()
{
        int value;
        List list;

        while (cout<<"Enter Value ( 0 to stop ) : ", cin >> value , value)
        {
                list.push_Back(value);
        }

        while (cout<<"Enter Value ( 0 to stop ) : ", cin >> value , value)
        {
                list.push_Front(value);
        }

        for (List :: Iterator lit = list.begin() ; lit != list.end() ; lit++)
                cout<<*lit<<endl;

        //for (auto i : list)

}
```