Implement Iterator class in class DLinkedList.

**Note**: Iterator is a concept of repetitive elements on sequence structures. Iterator is implemented in class vector, list in STL container in C++ (<https://www.geeksforgeeks.org/iterators-c-stl/>). Your task is to implement the simple same class with iterator in C++ STL container.

template <class T>  
class DLinkedList  
{  
public:  
    class Iterator; //forward declaration  
    class Node;     //forward declaration  
protected:  
    Node \*head;  
    Node \*tail;  
    int count;  
public:  
    DLinkedList() : head(NULL), tail(NULL), count(0){};  
    ~DLinkedList();  
    void add(const T &e);  
    void add(int index, const T &e);  
    T removeAt(int index);  
    bool removeItem(const T &item);  
    bool empty();  
    int size();  
    void clear();  
    T get(int index);  
    void set(int index, const T &e);  
    int indexOf(const T &item);  
    bool contains(const T &item);  
    string toString();  
    Iterator begin()  
    {  
        return Iterator(this, true);  
    }  
    Iterator end()  
    {  
        return Iterator(this, false);  
    }  
public:  
    **class Node**

**{**

**private:**

**T data;**

**Node \*next;**

**Node \*previous;**

**friend class DLinkedList<T>;**

**Iterator begin()**

**{**

**return Iterator(this, true);**

**}**

**Iterator end()**

**{**

**return Iterator(this, false);**

**}**

**public:**

**Node()**

**{**

**this->previous = NULL;**

**this->next = NULL;**

**}**

**Node(const T &data)**

**{**

**this->data = data;**

**this->previous = NULL;**

**this->next = NULL;**

**}**

**};**  
    class Iterator  
    {  
    private:  
        DLinkedList<T> \*pList;  
        Node \*current;

int index; // is the index of current in pList  
    public:  
        Iterator(DLinkedList<T> \*pList, bool begin);  
        Iterator &operator=(const Iterator &iterator);  
        void set(const T &e);  
        T &operator\*();  
        bool operator!=(const Iterator &iterator);

void remove();  
          
        // Prefix ++ overload  
        Iterator &operator++();  
          
        // Postfix ++ overload  
        Iterator operator++(int);  
    };  
};

**Please read example carefully to see how we use the iterator.**

**For example:**

| **Test** | **Result** |
| --- | --- |
| DLinkedList<int> list;  int size = 10;  for(int idx=0; idx < size; idx++){  list.add(idx);  }  DLinkedList<int>::Iterator it = list.begin();  for(; it != list.end(); it++)  {  cout << \*it << " |";  } | 0 |1 |2 |3 |4 |5 |6 |7 |8 |9 | |
| DLinkedList<int> list;  int size = 10;  for (int idx = 0; idx < size; idx++)  {  list.add(idx);  }  DLinkedList<int>::Iterator it = list.begin();  while (it != list.end())  {  it.remove();  it++;  }  cout << list.toString(); | [] |
| DLinkedList<int> list;  int size = 10;  for (int idx = 0; idx < size; idx++)  {  list.add(idx);  }  DLinkedList<int>::Iterator it = list.begin();  for(; it != list.end(); it++)  {  it.remove();  }  cout << list.toString(); | [] |