Implement Iterator class in class SLinkedList.

**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.

**Please read example and testcase carefully.**

template <class T>  
class SLinkedList  
{  
public:  
    class Iterator; //forward declaration  
    class Node;     //forward declaration  
protected:  
    Node \*head;  
    Node \*tail;  
    int count;  
public:  
    SLinkedList() : head(NULL), tail(NULL), count(0){};  
    ~SLinkedList();  
    void add(const T &e);  
    void add(int index, const T &e);  
    T removeAt(int index);  
    bool removeItem(const T &removeItem);  
    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();  
    SLinkedList(const SLinkedList &list)  
    {  
        this->count = 0;  
        this->head = NULL;  
        this->tail = NULL;  
    }  
    Iterator begin()  
    {  
        return Iterator(this, true);  
    }  
    Iterator end()  
    {  
        return Iterator(this, false);  
    }  
public:  
    class Node  
    {  
    private:  
        T data;  
        Node \*next;  
        friend class SLinkedList<T>;  
    public:  
        Node()  
        {  
            next = 0;  
        }  
        Node(Node \*next)  
        {  
            this->next = next;  
        }  
        Node(T data, Node \*next = NULL)  
        {  
            this->data = data;  
            this->next = next;  
        }  
    };  
    class Iterator  
    {  
    private:  
        SLinkedList<T> \*pList;  
        Node \*current;  
        int index; // corresponding with current node  
    public:  
        Iterator(SLinkedList<T> \*pList, bool begin);  
        Iterator &operator=(const Iterator &iterator);  
        void set(const T &e);  
        T &operator\*();  
        bool operator!=(const Iterator &iterator);  
          
        // Prefix ++ overload  
        Iterator &operator++();  
          
        // Postfix ++ overload  
        Iterator operator++(int);  
    };  
};

**For example:**

| **Test** | **Result** |
| --- | --- |
| SLinkedList<int> list;    int size = 10;  for(int idx=0; idx < size; idx++){  list.add(idx);  }    SLinkedList<int>::Iterator it;  int expvalue = 0;  for(it = list.begin(); it != list.end(); it++){  assert(\*it == expvalue);  expvalue += 1;  } |  |
| SLinkedList<int> list;    int size = 10;  for(int idx=0; idx < size; idx++){  list.add(idx);  }    SLinkedList<int>::Iterator it;  int expvalue = 0;  for(it = list.begin(); it != list.end(); ++it){  assert(\*it == expvalue);  expvalue += 1;  } |  |