Implement methods**get, set, empty, indexOf, contains**in template class **SLinkedList (which implements List ADT)**representing the singly linked list with type T with the initialized frame. The description of each method is given in the code.

template <class T>  
class SLinkedList {  
public:  
    class Node; // Forward declaration  
protected:  
    Node\* head;  
    Node\* tail;  
    int count;  
public:  
    SLinkedList(): head(NULL), tail(NULL), count(0);  
    ~SLinkedList() { };  
 void add(T e);  
 void add(int index, T e);  
 int size();  
 bool    empty();  
    T       get(int index);  
    void    set(int index, T e);  
    int     indexOf(T item);  
    bool    contains(T item);  
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;  
        }  
    };  
};

**For example:**

| **Test** | **Result** |
| --- | --- |
| SLinkedList<int> list;  int values[] = {10, 15, 2, 6, 4, 7, 40, 8};  int index[] = {0, 0, 1, 3, 2, 3, 5, 0};  int expvalues[]= {8, 15, 2, 4, 7, 10, 40, 6};  for (int idx = 0; idx < 8; idx++){  list.add(index[idx], values[idx]);  }  assert( list.size() == 8 );    for (int idx = 0; idx < 8; idx++){  assert( list.get(idx) == expvalues[idx] );  }  cout << list.toString(); | [8,15,2,4,7,10,40,6] |
| SLinkedList<int> list;  assert( list.empty() == true );  cout << list.toString(); | [] |