Implement methods**get, set, empty, indexOf, contains**in template class D**LinkedList (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 DLinkedList {  
public:  
    class Node; // Forward declaration  
protected:  
    Node\* head;  
    Node\* tail;  
    int count;  
public:  
    DLinkedList();  
    ~DLinkedList();  
 void add(const T &e);  
 void add(int index, const T &e);  
 int size();  
 bool empty();

T get(int index);

void set(int index, const T &e);

int indexOf(const T &item);

bool contains(const T &item);

public:

class Node

{

private:

T data;

Node \*next;

Node \*previous;

friend class DLinkedList<T>;

public:

Node()

{

this->previous = NULL;

this->next = NULL;

}

Node(const T &data)

{

this->data = data;

this->previous = NULL;

this->next = NULL;

}

};

};

In this exercise, we have include <iostream>, <string>, <sstream> and using namespace std.

**For example:**

| **Test** | **Result** |
| --- | --- |
| DLinkedList<int> list;  int size = 10;  for(int idx=0; idx < size; idx++){  list.add(idx);  }  for(int idx=0; idx < size; idx++){  cout << list.get(idx) << " |";  } | 0 |1 |2 |3 |4 |5 |6 |7 |8 |9 | |
| DLinkedList<int> list;  int size = 10;  int value[] = {2,5,6,3,67,332,43,1,0,9};  for(int idx=0; idx < size; idx++){  list.add(idx);  }  for(int idx=0; idx < size; idx++){  list.set(idx, value[idx]);  }  cout << list.toString(); | [2,5,6,3,67,332,43,1,0,9] |