Câu hỏi 1

Không hoàn thành

Chấm điểm của 1,00

Implement methods **add, size** in template class **DLinkedList (which implements List ADT)** representing the doubly 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();
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);`
`}`
`cout << list.toString();`
	`[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(0, idx);`
`}`
`cout << list.toString();`
	`[9,8,7,6,5,4,3,2,1,0]`

Answer: (penalty regime: 0, 0, 0, 5, 10 %)

Reset answer

```
1  template <class T>
2  void DLinkedList<T>::add(const T& e) {
3      /* Insert an element into the end of the list. */
4
5  }
6
7  template<class T>
8  void DLinkedList<T>::add(int index, const T& e) {
9      /* Insert an element into the list at given index. */
10
11 }
12
13 template<class T>
```

```
int DLinkedList<T>::size() {
    /* Return the length (size) of list */
    return 0;
}
```

Precheck Kiểm tra

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn

Câu hỏi 2

Không hoàn thành

Chấm điểm của 1,00

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]`

Answer: (penalty regime: 0, 0, 0, 5, 10 %)

Reset answer

```
1  template<class T>
2  T DLinkedList<T>::get(int index) {
```

```
    /* Give the data of the element at given index in the list. */

}

template <class T>
void DLinkedList<T>::set(int index, const T& e) {
    /* Assign new value for element at given index in the list */
}

template<class T>
bool DLinkedList<T>::empty() {
    /* Check if the list is empty or not. */

}

template<class T>
int DLinkedList<T>::indexOf(const T& item) {
    /* Return the first index wheter item appears in list, otherwise return -1 */

}

template<class T>
bool DLinkedList<T>::contains(const T& item) {
    /* Check if item appears in the list */

}
```

Precheck Kiểm tra

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn

Câu hỏi 3

Không hoàn thành

Chấm điểm của 1,00

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 |`

Test	Result

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();
	[]

Answer: (penalty regime: 0, 0, 0, 5, 10 %)

Reset answer

```
1 /*
2  * TODO: Implement class Iterator's method
3  * Note: method remove is different from SLinkedList, which is the advantage of DLir
```

```
 */
template <class T>
DLinkedList<T>::Iterator::Iterator(DLinkedList<T> *pList, bool begin)
{

}

template <class T>
typename DLinkedList<T>::Iterator& DLinkedList<T>::Iterator::operator=(const DLinkec
{

}

template <class T>
void DLinkedList<T>::Iterator::set(const T &e)
{

}

template<class T>
T& DLinkedList<T>::Iterator::operator*()
{

}

template<class T>
void DLinkedList<T>::Iterator::remove()
{
    /*
    * TODO: delete Node in pList which Node* current point to.
    *       After that, Node* current point to the node before the node just deletec
    *       If we remove first node of pList, Node* current point to nullptr.
    *       Then we use operator ++, Node* current will point to the head of pList.
    */
}

template<class T>
bool DLinkedList<T>::Iterator::operator!=(const DLinkedList::Iterator &iterator)
{

}
```

```
46 template<class T>
47 typename DLinkedList<T>::Iterator& DLinkedList<T>::Iterator::operator++()
48 {
49
50 }
51
52 template<class T>
53 typename DLinkedList<T>::Iterator DLinkedList<T>::Iterator::operator++(int)
54 {
55
56 }
57
58
```

Precheck Kiểm tra

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn

Câu hỏi 4

Không hoàn thành

Chấm điểm của 1,00

Implement methods **removeAt, removeItem, clear** 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 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);
    T       removeAt(int index);
    bool    removeItem(const T &item);
    void    clear();
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;
int value[] = {2,5,6,3,67,332,43,1,0,9};
 	
for(int idx=0; idx < size; idx++){
  list.add(value[idx]);
}
list.removeAt(0);
cout << list.toString();
	[5,6,3,67,332,43,1,0,9]

Answer: (penalty regime: 0 %)

Reset answer

```
1  template <class T>
2  T DLinkedList<T>::removeAt(int index)
3  {
4      /* Remove element at index and return removed value */
5  }
6
7  template <class T>
8  bool DLinkedList<T>::removeItem(const T& item)
9  {
10     /* Remove the first appearance of item in list and return true, otherwise return
```

```
11
12  }
13
14  template<class T>
15  void DLinkedList<T>::clear(){
16      /* Remove all elements in list */
17  }
18
```

Kiểm tra

Câu hỏi 5

Không hoàn thành

Chấm điểm của 1,00

In this exercise, we will use Standard Template Library List (click open in other tab to show more) to implement a Data Log.

This is a simple implementation in applications using undo and redo. For example in Microsoft Word, you must have nodes to store states when Ctrl Z or Ctrl Shift Z to go back or forward.

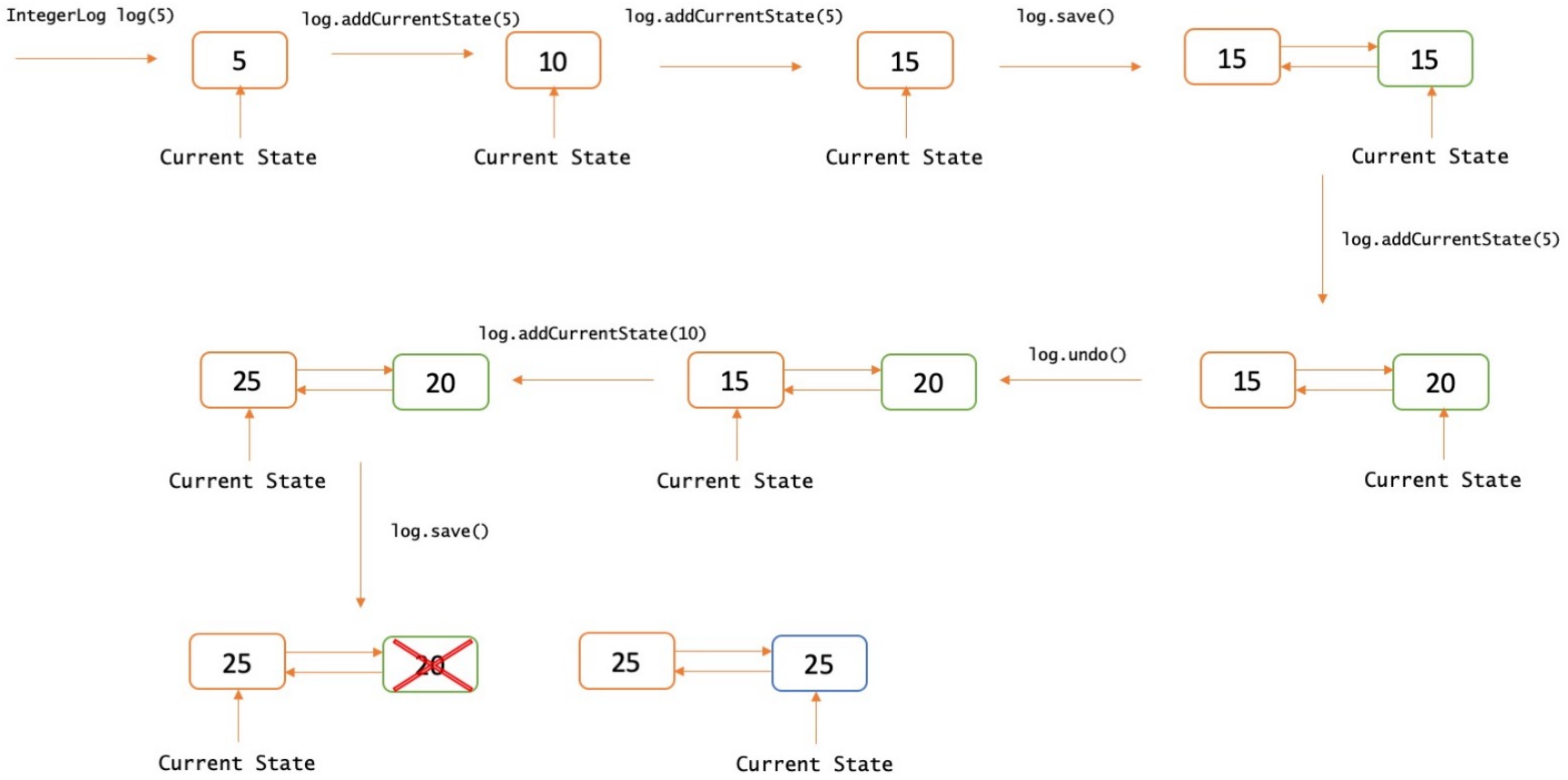
DataLog has a doubly linked list to store the states of data (an integer) and iterator to mark the current state. Each state is stored in a node, the transition of states is depicted in the figure below.

Your task in this exercise is implement functions marked with /* * TODO */.

```
class DataLog
{
private:
    list<int> logList;
    list<int>::iterator currentState;

public:
    DataLog();
    DataLog(const int &data);
    void addCurrentState(int number);
    void subtractCurrentState(int number);
    void save();
    void undo();
    void redo();

    int getCurrentStateData()
    {
        return *currentState;
    }

    void printLog()
    {
        for (auto i = logList.begin(); i != logList.end(); i++) {
            if(i == currentState) cout << "Current state: ";
            cout << "[ " << *i << " ] => ";
        }
        cout << "END_LOG";
    }
};
```

Note: Normally, when we say a List, we talk about doubly linked list. For implementing a singly linked list, we use forward list.

We have include <iostream> <list> and using namespace std;



For example:

Test	Result

DataLog log(10);
log.save();
log.addCurrentState(15);
log.save();
log.addCurrentState(15);
log.undo();
log.printLog();
	[10] => Current state: [25] => [40] => END_LOG

Test	Result

DataLog log(10);
log.save();
log.addCurrentState(15);
log.save();
log.addCurrentState(15);
log.save();
log.subtractCurrentState(5);
log.printLog();
	[10] => [25] => [40] => Current state: [35] => END_LOG

Answer: (penalty regime: 0, 0, 0, 5, 10 %)

Reset answer

```
DataLog::DataLog()
{
    /*
     * TODO:  add the first state with 0
     */
}

DataLog::DataLog(const int &data)
{
    /*
     * TODO:  add the first state with data
     */
}

void DataLog::addCurrentState(int number)
{
    /*
     * TODO: Increase the value of current state by number
     */
}

void DataLog::subtractCurrentState(int number)
{
    /*
```

```
26      */
27  }
28
29  void DataLog::save()
30  {
31      /*
32       * TODO: This function will create a new state, copy the data of the currentS
33       *       and move the currentState Iterator to this new state. If there are o
34       *       currentState Iterator, we delete them all before creating a new stat
35       */
36  }
37
38  void DataLog::undo()
39  {
40      /*
41       * TODO: Switch to the previous state of the data
42       *       If this is the oldest state in the log, nothing changes
43       */
44  }
45
46  void DataLog::redo()
47  {
48      /*
49
```

Precheck Kiểm tra

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn

Câu hỏi 6

Không hoàn thành

Chấm điểm của 1,00

Given the head of a doubly linked list, two positive integer a and b where a <= b. Reverse the nodes of the list from position a to position b and return the reversed list

Note: the position of the first node is 1. It is guaranteed that a and b are valid positions. You MUST NOT change the val attribute in each node.

```
struct ListNode {
    int val;
    ListNode *left;
    ListNode *right;
    ListNode(int x = 0, ListNode *l = nullptr, ListNode* r = nullptr) : val(x), left(l), right(r) {}
};
```

Constraint:
1 <= list.length <= 10^5
0 <= node.val <= 5000
1 <= left <= right <= list.length

Example 1:
Input: list = {3, 4, 5, 6, 7} , a = 2, b = 4
Output: 3 6 5 4 7

Example 2:
Input: list = {8, 9, 10}, a = 1, b = 3
Output: 10 9 8

For example:

Test	Input	Result

`int size;`
`cin >> size;`
`int* list = new int[size];`
`for(int i = 0; i < size; i++) {`
`cin >> list[i];`
`}`
`int a, b;`
`cin >> a >> b;`
`unordered_map<ListNode*, int> nodeValue;`
`ListNode* head = init(list, size, nodeValue);`
`ListNode* reversed = reverse(head, a, b);`
`try {`
`printList(reversed, nodeValue);`
`}`
`catch(char const* err) {`
`cout << err << '\n';`
`}`
`freeMem(head);`
`delete[] list;`
	5
	3 4 5 6 7
	2 4
		3 6 5 4 7

Test	Input	Result

int size;
cin >> size;
int* list = new int[size];
for(int i = 0; i < size; i++) {
cin >> list[i];
}
int a, b;
cin >> a >> b;
unordered_map<ListNode*, int> nodeValue;
ListNode* head = init(list, size, nodeValue);
ListNode* reversed = reverse(head, a, b);
try {
printList(reversed, nodeValue);
}
catch(char const* err) {
cout << err << '\n';
}
freeMem(head);
delete[] list;
	3
	8 9 10
	1 3
		10 9 8

Answer: (penalty regime: 0 %)

Reset answer

```
/*
struct ListNode {
    int val;
    ListNode *left;
    ListNode *right;
    ListNode(int x = 0, ListNode *l = nullptr, ListNode* r = nullptr) : val(x), left
};
*/

ListNode* reverse(ListNode* head, int a, int b) {
    /To Do
}
```

Precheck Kiểm tra

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn