Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 2 tháng 10 2024, 8:05 AM
Kết thúc lúc	Chủ Nhật, 13 tháng 10 2024, 12:39 PM
Thời gian thực hiện	11 Các ngày 4 giờ
Điểm	0,00/5,00
Điểm	**0,00** trên 10,00 (**0**%)

Câu hỏi **1**

Sai

Đạt điểm 0,00 trên 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

```
template <class T>
void DLinkedList<T>::add(const T& e) {
    /* Insert an element into the end of the list. */

}

template<class T>
void DLinkedList<T>::add(int index, const T& e) {
    /* Insert an element into the list at given index. */

}

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

	Test	Expected	Got	

✗	`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]`	`[]`	✗

	Test	Expected	Got	

✗	`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]`	`[]`	✗

Some hidden test cases failed, too.

Show differences

Sai

Marks for this submission: 0,00/1,00.

Câu hỏi **2**

Sai

Đạt điểm 0,00 trên 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

```
template<class T>
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 */

}
```

Syntax Error(s)

```
__tester__.cpp: In member function 'bool DLinkedList<T>::empty()':
__tester__.cpp:148:1: error: no return statement in function returning non-void [-Werror=return-type]
  148 | }
      | ^
__tester__.cpp: In member function 'int DLinkedList<T>::indexOf(const T&)':
__tester__.cpp:154:1: error: no return statement in function returning non-void [-Werror=return-type]
  154 | }
      | ^
__tester__.cpp: In member function 'bool DLinkedList<T>::contains(const T&)':
__tester__.cpp:160:1: error: no return statement in function returning non-void [-Werror=return-type]
  160 | }
      | ^
__tester__.cpp: In instantiation of 'T DLinkedList<T>::get(int) [with T = int]':
__tester__.cpp:170:23:   required from here
__tester__.cpp:137:1: error: no return statement in function returning non-void [-Werror=return-type]
  137 | }
      | ^
cc1plus: all warnings being treated as errors
```

Sai

Marks for this submission: 0,00/1,00.

Câu hỏi **3**

Sai

Đạt điểm 0,00 trên 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

```
template <class T>
T DLinkedList<T>::removeAt(int index)
{
    /* Remove element at index and return removed value */
}

template <class T>
bool DLinkedList<T>::removeItem(const T& item)
{
    /* Remove the first apperance of item in list and return true, otherwise return false */

}

template<class T>
void DLinkedList<T>::clear(){
    /* Remove all elements in list */
}

```

Syntax Error(s)

```
__tester__.cpp: In member function 'bool DLinkedList<T>::removeItem(const T&)':
__tester__.cpp:237:1: error: no return statement in function returning non-void [-Werror=return-type]
  237 | }
      | ^
__tester__.cpp: In instantiation of 'T DLinkedList<T>::removeAt(int) [with T = int]':
__tester__.cpp:254:16:   required from here
__tester__.cpp:230:1: error: no return statement in function returning non-void [-Werror=return-type]
  230 | }
      | ^
cc1plus: all warnings being treated as errors
```

Sai

Marks for this submission: 0,00/1,00.

Câu hỏi **4**

Sai

Đạt điểm 0,00 trên 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.

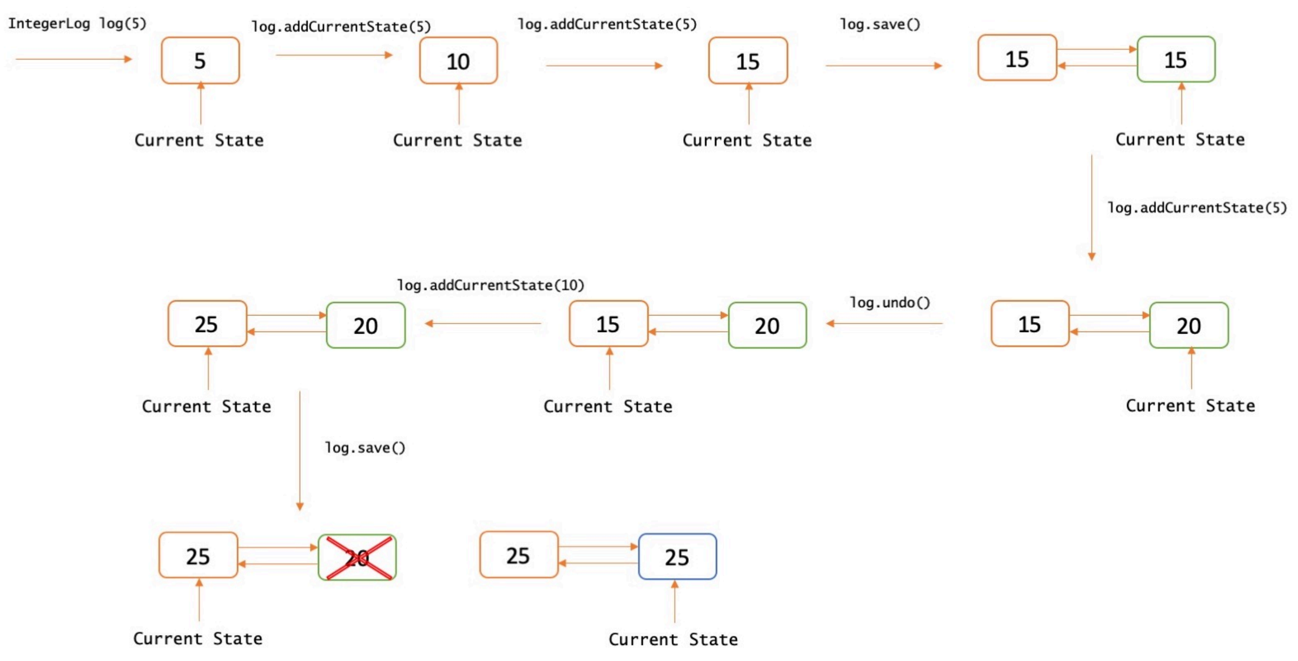
DataLog has a doubly linked list to store the states of data (an integer) and iterator to mark the current state. Each state is stored in a node, the transition of states is depicted in the figure below.

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`

`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)
{
    /*
     * TODO: Decrease the value of current state by number
     */
}

void DataLog::save()
{
    /*
     * TODO: This function will create a new state, copy the data of the currentState
     *       and move the currentState Iterator to this new state. If there are other states behind the
     *       currentState Iterator, we delete them all before creating a new state.
     */
}

void DataLog::undo()
{
    /*
     * TODO: Switch to the previous state of the data
     *       If this is the oldest state in the log, nothing changes
     */
}

void DataLog::redo()
{
    /*
     * TODO: Switch to the latter state of the data
     *       If this is the latest state in the log, nothing changes
     */
```

	Test	Expected	Got	

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

Testing was aborted due to error.

Show differences

Sai

Marks for this submission: 0,00/1,00.

Câu hỏi **5**

Sai

Đạt điểm 0,00 trên 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(l), right(r) {}
};
*/

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

Syntax Error(s)

```
__tester__.cpp: In function 'ListNode* reverse(ListNode*, int, int)':
__tester__.cpp:73:5: error: expected primary-expression before '/' token
   73 |     /To Do
      |     ^
__tester__.cpp:73:6: error: 'To' was not declared in this scope
   73 |     /To Do
      |      ^~
__tester__.cpp:74:1: error: no return statement in function returning non-void [-Werror=return-type]
   74 | }
      | ^
cc1plus: all warnings being treated as errors
```

Sai

Marks for this submission: 0,00/1,00.