



BÁCH KHOA E-LEARNING

[Trang của tôi](#) / [Khóa học](#) / [Học kỳ II năm học 2020-2021 \(Semester 2 - Academic year 2020-2021\)](#)

/ [Chương Trình Chất Lượng Cao dạy bằng Tiếng Anh \(High-Quality training program\)](#)

/ [Khoa Khoa học và Kỹ thuật Máy tính \(Faculty of Computer Science and Engineering\)](#) / [Khoa Học Máy Tính](#)

/ [Data Structures and Algorithms \(Lab\) \(CO2004\) _Bảng Ngọc Bảo Tâm \(CC_HK202\)](#) / [Link list](#) / [Exercises](#)

Đã bắt đầu vào lúc Saturday, 17 April 2021, 9:57 AM

Tình trạng Đã hoàn thành

Hoàn thành vào lúc Saturday, 24 April 2021, 3:27 AM

Thời gian thực hiện 6 ngày 17 giờ

Điểm 2,90 của 3,00 (97%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Implement methods **add**, **size** 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();
    ~SLinkedList();
    void add(T e);
    void add(int index, T e);
    int size();
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) {
            this->data = data;
            this->next = next;
        }
    };
};
```

For example:

Test	Result
<pre>SLinkedList<int> list; int size = 10; for(int index = 0; index < size; index++){ list.add(index); } cout << list.toString();</pre>	[0,1,2,3,4,5,6,7,8,9]

Answer: (penalty regime: 0 %)

Reset answer

```
1 template <class T>
2 void SLinkedList<T>::add(const T& e) {
3     // allocating space
4     T data=e;
5     Node* newNode = new Node();
6
7     // inserting the required data
```

```

8      newNode->data = data;
9      newNode->next = NULL;
10
11     // If the Linked List is empty, then make the new node as head
12 if (head == NULL)
13 {
14     //////////////////////////////////////
15     head = newNode;
16     tail = newNode;
17 }
18 else {
19     tail->next=newNode;
20     tail=newNode;}
21 count++;
22 }
23 //////////////////////////////////////
24 template <class T>
25 void SLinkedList<T>::add(int index, const T& e) {
26
27     int pos = index; //?//
28     Node** current = &head;
29     T data = e;
30
31     if (pos > count ) pos = count;
32     if (pos < 0) pos = 0;
33
34     // allocating space
35     Node* newNode = new Node();
36
37     // inserting the required data
38     newNode->data = data;
39     newNode->next = NULL;
40
41     // allocating space
42     Node* temp = new Node();
43
44     // inserting the required data
45     temp->data = data;
46     temp->next = NULL;
47
48
49     // If the Linked List is empty, then make the new node as head
50 if (head == NULL)
51 {
52     //////////////////////////////////////
53     head = newNode;
54     tail = newNode;
55 }
56
57 if (pos == 0)
58 {
59     newNode->next = head; //link to old head
60     head = newNode; //then become new head
61 }
62 else if (pos == count) {
63     add(data);
64     count--;
65 }
66
67 else {
68
69     // Keep looping until the pos is zero
70     for(int i=0;i<pos;i++) {
71         current = &(*current)->next;
72     }
73     temp->next = *current;
74     *current = temp;
75 }
76
77 count++;
78 }
79
80 template<class T>
81 int SLinkedList<T>::size() {
82     if (head == NULL) count =0;
83
84     return count;
85 }
86

```

	Test	Expected	Got	
✓	<pre> SLinkedList<int> list; int size = 10; for(int index = 0; index < size; index++){ list.add(index); } cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓
✓	<pre> SLinkedList<int> list; int size = 10; for(int index = 0; index < size; index++){ list.add(0, index); } cout << list.toString(); </pre>	[9,8,7,6,5,4,3,2,1,0]	[9,8,7,6,5,4,3,2,1,0]	✓
✓	<pre> SLinkedList<int> list; int size = 10; for (int index = 0; index < size; index++) { list.add(list.size(), index); } cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓
✓	<pre> SLinkedList<int> list; int values[] = {10, 15, 2, 6, 4, 7, 40, 8}; int index[] = {0, 0, 1, 3, 2, 3, 5, 0}; for (int idx = 0; idx < 8; idx++){ list.add(index[idx], values[idx]); } cout << list.toString(); </pre>	[8,15,2,4,7,10,40,6]	[8,15,2,4,7,10,40,6]	✓
✓	<pre> SLinkedList<int> list; int values[] = {10, 15, 2, 6, 4, 7, 40, 8}; int index[] = {0, 0, 1, 3, 2, 3, 5, 0}; for (int idx = 0; idx < 8; idx++){ list.add(index[idx], values[idx]); } cout << list.size(); </pre>	8	8	✓
✓	<pre> SLinkedList<int> list; list.add(0); for (int i= 0; i< 10; i++) { list.add(list.size() - 1, list.size()); } cout << list.toString(); </pre>	[1,2,3,4,5,6,7,8,9,10,0]	[1,2,3,4,5,6,7,8,9,10,0]	✓
✓	<pre> SLinkedList<int> list; int values[] = { 13, 23, 7, 9, 8, 7, 50, -1 }; int index[] = { 0, 1, 1, 3, 2, 3, 5, 1 }; for (int idx = 0; idx < 8; idx++) { list.add(index[idx], values[idx]); } cout << list.toString(); </pre>	[13,-1,7,8,7,23,50,9]	[13,-1,7,8,7,23,50,9]	✓

	Test	Expected	Got	
✓	<pre> SLinkedList<int> list; int values[] = { 13, 23, 7, 9, 8, 7, 50, -1 }; int index[] = { 0, 1, 1, 3, 2, 3, 5, 1 }; for (int idx = 0; idx < 8; idx++) { list.add(index[idx], values[idx]); } cout << list.size(); </pre>	8	8	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; i++) { if (i % 2 == 0) list.add(0, i); else list.add(list.size(), i); } cout << list.size(); </pre>	10	10	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; i++) { if (i % 2 == 0) list.add(0, i); else list.add(list.size(), i); } cout << list.toString(); </pre>	[8,6,4,2,0,1,3,5,7,9]	[8,6,4,2,0,1,3,5,7,9]	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **2**

Chính xác

Điểm 1,00 của 1,00

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
------	--------

Test	Result
<pre> 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(); </pre>	[8,15,2,4,7,10,40,6]

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```

1  template<class T>
2  T SLinkedList<T>::get(int index) {
3      /* Give the data of the element at given index in the list. */
4
5      Node* tmp=new Node;
6      tmp=head;
7      if(index<0||index>(count-1)){
8          throw std::out_of_range("F");
9      }
10     for(int i = 0; i< index; i++)    tmp=tmp->next;
11     return tmp->data;
12
13 }
14
15
16
17
18 template <class T>
19 void SLinkedList<T>::set(int index, const T& e) {
20     /* Assign new value for element at given index in the list */
21     Node* tmp=new Node;
22     tmp=head;
23     if(index<0||index>(count-1)){
24         throw std::out_of_range("F");
25     }
26     for(int i=0;i<index;i++){
27         tmp=tmp->next;
28     }
29     tmp->data=e;
30
31
32
33
34 template<class T>
35 bool SLinkedList<T>::empty() {
36     /* Check if the list is empty or not. */
37     if(head==NULL)return true;
38     else return false;
39 }
40
41
42
43 template<class T>
44 int SLinkedList<T>::indexOf(const T& item) {
45     /* Return the first index wheter item appears in list, otherwise return -1 */
46     Node* temp = head;
47     int coun = 0;
48     while(temp!= NULL)
49     {
50         if (temp->data == item) return coun;
51         ++coun;
52         temp = temp->next;
53     }
54     //return -1 if no match

```

```

54 //return -1 if no match
55 return -1;
56 }
57
58
59
60 template<class T>
61 bool SLinkedList<T>::contains(const T& item) {
62     Node* temp = head;
63     while(temp!= NULL)
64     {
65         if (temp->data == item) return 1;
66         temp = temp->next;
67     }
68     return 0;
69 }

```

	Test	Expected	Got	
✓	<pre> 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(); </pre>	[8,15,2,4,7,10,40,6]	[8,15,2,4,7,10,40,6]	✓
✓	<pre> SLinkedList<int> list; assert(list.empty() == true); cout << list.toString(); </pre>	[]	[]	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } assert(list.empty() == false); cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } for (int i = 0; i < 10; ++i) { assert(list.indexOf(i) == i); } cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓

	Test	Expected	Got	
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } for (int i = 0; i < 10; ++i) { assert(list.contains(i) == true); } cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } for (int i = 10; i < 20; ++i) { assert(list.indexOf(i) == -1); } cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } for (int i = 10; i < 20; ++i) { assert(list.contains(i) == false); } cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8,9]	[0,1,2,3,4,5,6,7,8,9]	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } for (int i = 0; i < 10; ++i) { list.set(i, i + 10); } for (int i = 0; i < 10; ++i) { assert(list.get(i) == i + 10); } cout << list.toString(); </pre>	[10,11,12,13,14,15,16,17,18,19]	[10,11,12,13,14,15,16,17,18,19]	✓
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } try { list.get(100); } catch(std::out_of_range e){ assert(1 == 1); //pass e.what(); } </pre>			✓

	Test	Expected	Got	
✓	<pre>SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } try { list.set(100, 100); } catch (std::out_of_range e) { assert(1 == 1); //pass e.what(); }</pre>			✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Đúng một phần

Điểm 0,90 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 SLinkedList {
public:
    class Node; // Forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    SLinkedList();
    ~SLinkedList();
    void add(T e);
    void add(int index, T e);
    int size();
    bool empty();
    int size();
    void clear();
    T get(int index);
    void set(int index, T e);
    int indexOf(T item);
    bool contains(T item);
    T removeAt(int index);
    bool removeItem(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
<pre>SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } assert(list.get(0) == list.removeAt(0)); cout << list.toString();</pre>	[1,2,3,4,5,6,7,8,9]

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```

1 |
2 |
3 | //////////////////////////////////////////////////
4 | template <class T>
5 | T SLinkedList<T>::removeAt(int index)
6 | {
7 |     /* Remove element at index and return removed value */
8 |     T tdata;
9 |     // If linked list is empty
10 |    if (head == NULL) throw "nothing to delete";
11 |
12 |    // Store head node
13 |    Node* temp = head;
14 |
15 |    // If head needs to be removed
16 |    if (index == 0)
17 |    {
18 |
19 |        // Change head
20 |        head = temp->next;
21 |
22 |        // Free old head
23 |        T tdata = temp->data;
24 |        delete(temp);
25 |        --count;
26 |        return tdata;
27 |    }
28 |
29 |    // Find previous node of the node to be deleted
30 |    for (int i = 0; temp != NULL && i < index - 1; i++)
31 |        temp = temp->next;
32 |
33 |    // If index is more than number of nodes
34 |    // if (temp == NULL || temp->next == NULL) throw "out of range";
35 |
36 |    // Node temp->next is the node to be deleted
37 |    // Store pointer to the next of node to be deleted
38 |    Node* next = temp->next->next;
39 |
40 |    // Unlink the node from linked list
41 |    tdata = temp->next->data;
42 |    delete(temp->next); // Free memory
43 |
44 |    // Unlink the deleted node from list
45 |    temp->next = next;
46 |    --count;
47 |    return tdata;
48 | }
49 |
50 | template <class T>
51 | bool SLinkedList<T>::removeItem(const T& item)
52 | {
53 |     /* Remove the first apperance of item in list and return true, otherwise return false */
54 |
55 |     int i;
56 |     /* Remove the first apperance of item in list and return true, otherwise return false */
57 |     i = SLinkedList::indexOf(item);
58 |     if (i == -1) return false;
59 |     else
60 |     {
61 |         removeAt(i);
62 |         return true;
63 |     }
64 | }
65 |
66 |
67 |
68 |
69 | template<class T>
70 | void SLinkedList<T>::clear() {
71 |     /* Remove all elements in list */
72 |     for (int i = 0; i < count; i++) {
73 |         removeAt(i);
74 |     }
75 |     count = 0;
76 | }

```

	Test	Expected	Got	
✓	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } assert(list.get(0) == list.removeAt(0)); cout << list.toString(); </pre>	[1,2,3,4,5,6,7,8,9]	[1,2,3,4,5,6,7,8,9]	✓
✗	<pre> SLinkedList<int> list; for (int i = 0; i < 10; ++i) { list.add(i); } assert(list.get(9) == list.removeAt(9)); cout << list.toString(); </pre>	[0,1,2,3,4,5,6,7,8]	***Error*** Segmentation fault (core dumped)	✗

Testing was aborted due to error.

Show differences

Đúng một phần

Điểm cho bài nộp này: 0,10/1,00. Tính toán cho lần làm bài trước đó, điểm **0,90/1,00**.

◀ Data Structures and Algorithms (Lab) - CC02 - 1PM to 3:50PM - 25/05/2021

Chuyển tới...

Test ▶

Copyright 2007-2020 BKĐT-Đại Học Bách Khoa Tp.HCM. All Rights Reserved.

Địa chỉ: Nhà A1- 268 Lý Thường Kiệt, Phường 14, Quận 10, Tp.HCM.

Email: elearning@hcmut.edu.vn

Phát triển dựa trên hệ thống Moodle