Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Năm, 27 tháng 2 2025, 3:56 PM
Kết thúc lúc	Thứ Năm, 27 tháng 2 2025, 5:00 PM
Thời gian thực hiện	1 giờ 4 phút
Điểm	80,00/80,00
Điểm	**10,00** trên 10,00 (**100**%)

Câu hỏi 1

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một `struct Node` biểu diễn một $node$ của 1 danh sách liên kết đơn như sau:

```
struct Node
{
    int value;
    Node* next;
};
```

Biết $head$ là con trỏ trỏ tới $node$ đầu tiên của danh sách liên kết, viết các hàm sau:

- `void print(Node* head);` in ra giá trị các phần tử của danh sách liên kết trên cùng một dòng, cách nhau bởi 1 dấu cách.
- `Node* insertHead(Node* head, int value);` thêm một $node$ có giá trị là $value$ vào đầu của danh sách liên kết. Hàm trả về con trỏ trỏ tới đầu danh sách liên kết.
- `Node* insertTail(Node* head, int value);` thêm một $node$ có giá trị là $value$ vào cuối của danh sách liên kết. Hàm trả về con trỏ trỏ tới đầu danh sách liên kết
- `Node* deleteNode(Node* head, int pos);` xoá một $node$ tại vị trí thứ pos ra khỏi danh sách liên kết, biết pos luôn lớn hơn hoặc bằng 0 và nhỏ hơn độ dài của danh sách liên kết. Hàm trả về con trỏ trỏ tới đầu danh sách liên kết.
- `int getValue(Node* head, int pos);` trả về giá trị của $node$ tại vị trí thứ pos biết pos luôn lớn hơn hoặc bằng 0 và nhỏ hơn độ dài của danh sách liên kết.

For example:

Test	Input	Result
`print(head);`	5 4 2 1 3 5	4 2 1 3 5
`head = insertHead(head, 5);` `print(head);`	5 4 2 1 3 5	5 4 2 1 3 5
`head = insertTail(head, 2);` `print(head);`	5 4 2 1 3 5	4 2 1 3 5 2
`head = deleteNode(head, 2);` `print(head);`	5 4 2 1 3 5	4 2 3 5
`cout << getValue(head, 2);`	5 4 2 1 3 5	1

Answer: (penalty regime: 0 %)

Reset answer

```
1  void print(Node* head)
2  {
3      // Your code here
4      Node *p = head;
5      while (p) {
6          cout << p->value << ' ';
7          p = p->next;
8      }
9  }
10
11 Node* insertHead(Node* head, int value)
12 {
13     // Your code here
14     Node *p = new Node;
15     p->value = value;
16     p->next = head;
17     head = p;
```

```cpp
    return head;
}

Node* insertTail(Node* head, int value)
{
    // Your code here
    Node *p = head;
    if (!p) {
        return insertHead(head, value);
    }
    while (p->next) {
        p = p->next;
    }
    Node *q = new Node;
    q->value = value;
    q->next = nullptr;
    p->next = q;
    return head;
}

Node* deleteNode(Node* head, int pos)
{
    // Your code here
    Node *p = head;
    if (!p->next) {
        delete p;
        return NULL;
    }
    for (int i = 0; i < pos - 1; ++i) {
        p = p->next;
    }
    Node *q = p->next;
    p->next = q->next;
    delete q;
    return head;
```

	Test	Input	Expected	Got	
✓	print(head);	5 4 2 1 3 5	4 2 1 3 5	4 2 1 3 5	✓
✓	head = insertHead(head, 5); print(head);	5 4 2 1 3 5	5 4 2 1 3 5	5 4 2 1 3 5	✓
✓	head = insertTail(head, 2); print(head);	5 4 2 1 3 5	4 2 1 3 5 2	4 2 1 3 5 2	✓
✓	head = deleteNode(head, 2); print(head);	5 4 2 1 3 5	4 2 3 5	4 2 3 5	✓
✓	cout << getValue(head, 2);	5 4 2 1 3 5	1	1	✓
✓	print(head);	0			✓
✓	print(head);	1 4	4	4	✓
✓	head = insertHead(head, 4); print(head);	0	4	4	✓
✓	head = insertHead(head, 4); print(head);	3 1 2 3	4 1 2 3	4 1 2 3	✓
✓	head = insertTail(head, 2); print(head);	0	2	2	✓
✓	head = insertTail(head, 4); print(head);	3 1 2 3	1 2 3 4	1 2 3 4	✓

	Test	Input	Expected	Got	
✓	head = deleteNode(head, 0); print(head);	1 3			✓
✓	head = deleteNode(head, 3); print(head);	4 3 2 1 4	3 2 1	3 2 1	✓
✓	head = deleteNode(head, 2); print(head);	4 3 2 1 4	3 2 4	3 2 4	✓
✓	cout << getValue(head, 0);	1 2	2	2	✓
✓	cout << getValue(head, 1);	4 4 2 3 1	2	2	✓
✓	cout << getValue(head, 3);	4 4 2 3 1	1	1	✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
void print(Node* head)
{
    // Your code here
    while (head != NULL) {
        cout << head->value << " ";
        head = head->next;
    }
}

Node* insertHead(Node* head, int value)
{
    // Your code here
    Node* new_node = new Node();
    new_node->value = value;
    new_node->next = head;
    return new_node;
}

Node* insertTail(Node* head, int value)
{
    // Your code here
    Node* new_node = new Node();
    new_node->value = value;
    if (head == NULL) return new_node;
    Node* i = head;
    for (; i->next != NULL; i=i->next);
    i->next = new_node;
    return head;
}

Node* deleteNode(Node* head, int pos)
{
    // Your code here
    if (pos == 0) {
        Node* next = head->next;
        delete head;
        return next;
    }
    Node* i = head;
    while (--pos) i = i->next;
    Node* deleted = i->next;
    i->next = deleted->next;
    delete deleted;
    return head;
}

int getValue(Node* head, int pos)
{
    // Your code here
    while (pos--) head = head->next;
    return head->value;
}
```

```
52 }
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 2

Đúng

Đạt điểm 10,00 trên 10,00

Cho một `struct Node` biểu diễn một $node$ của 1 danh sách liên kết như sau:

```
struct Node
{
    int value;
    Node *next;
};
```

Yêu cầu:

Viết hàm `bool compareLists(Node* headA, Node* headB);` so sánh 2 danh sách liên kết. Hai danh sách liên kết giống nhau khi có cùng độ dài và giá trị của các $node$ tại vị trí tương ứng bằng nhau.

Input:

- Hai con trỏ trỏ đến các node head của 2 danh sách.

Output:

- Hàm trả về $true$ nếu 2 danh sách liên kết giống nhau, nếu không trả về $false$.

Gợi ý:

- Dùng một vòng lặp duyệt lần lượt trên cả 2 danh sách liên kết.
- Nếu có một giá trị khác hoặc một trong hai danh sách hết phần tử trước thì trả về false.
- Cuối cùng, sau vòng lặp trả về true.

For example:

Test	Input	Result
`cout << compareLists(headA, headB);`	4 4 2 3 1 4 2 3 1 4	1

Answer: (penalty regime: 0 %)

Reset answer

```
bool compareLists(Node* headA, Node* headB) {
    // Your code here
    Node *p = headA;
    Node *q = headB;
    while (p && q) {
        if (p->value != q->value) {
            return false;
        }
        p = p->next;
        q = q->next;
    }
    return !p && !q; // Change this line
}
```

	Test	Input	Expected	Got	
✓	`cout << compareLists(headA, headB);`	4 4 2 3 1 4 2 3 1 4	1	1	✓

✓	`cout << compareLists(headA, headB);`
	`cout << compareLists(headB, headA);`
	`cout << compareLists(headA, headA);`
	`cout << compareLists(headB, headB);`
		4 4
		2 3 1 4
		2 3 2 4
			0011	0011	✓

✓	`cout << compareLists(headA, headB);`	1 0 3	0	0	✓
✓	`cout << compareLists(headA, headB);`	0 1 3	0	0	✓
✓	`cout << compareLists(headA, headB);` `cout << compareLists(headB, headA);`	4 3 3 3 3 3 3 3 3	00	00	✓
✓	`cout << compareLists(headA, headB);`	7 7 4 1 2 3 3 1 2 4 1 2 3 3 1 2	1	1	✓
✓	`cout << compareLists(headA, headB);`	0 0	1	1	✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
bool compareLists(Node* headA, Node* headB) {
    // Your code here
    while (headA !=NULL && headB != NULL) {
        if (headA->value != headB->value) return false;
        headA = headA->next;
        headB = headB->next;
    }
    return headA == headB; // both equal NULL
}
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 3

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một `struct Node` biểu diễn một *node* của 1 danh sách liên kết đơn như sau:

```
struct Node
{
    int value;
    Node* next;
};
```

Viết các hàm:

- `vector<int> linkedListToVector(Node* head)` chuyển đổi một danh sách liên kết thành một vector.
- `Node* vectorToLinkedList(vector<int> values)` chuyển đổi một vector thành một danh sách liên kết.

For example:

Test	Input	Result
`vector<int> v = linkedListToVector(head);` `print(v);`	5 1 5 2 4 7	1 5 2 4 7
`Node* ll = vectorToLinkedList(values);` `print(ll);`	5 1 5 2 4 7	1 5 2 4 7

Answer: (penalty regime: 0 %)

Reset answer

```
vector<int> linkedListToVector(Node* head)
{
    // Your code here
    vector<int> v;
    Node *p = head;
    while (p) {
        v.emplace_back(p->value);
        p = p->next;
    }
    return v; // Change this line
}

Node* vectorToLinkedList(vector<int> values)
{
    // Your code here
    Node *head = new Node;
    head->value = values[0];
    Node *p = head;
    for (int i = 1; i < int(values.size()); ++i) {
        Node *q = new Node;
        q->value = values[i];
        p->next = q;
        p = q;
    }
    return head; // Change this line
}
```

	Test	Input	Expected	Got	
✓	`vector<int> v = linkedListToVector(head);` `print(v);`	5 1 5 2 4 7	1 5 2 4 7	1 5 2 4 7	✓

	Test	Input	Expected	Got	
✓	`Node* ll = vectorToLinkedList(values);` `print(ll);`	5 1 5 2 4 7	1 5 2 4 7	1 5 2 4 7	✓

✓	`vector<int> v = linkedListToVector(head);`
	`cout << v.size() << endl;`
	`print(v);`
		1
		5
			1
			5
				1
				5
					✓

✓	`Node* ll = vectorToLinkedList(values);` `print(ll);`	1 5	5	5	✓
✓	`vector<int> v = linkedListToVector(head);` `print(v);`	10 4 2 1 3 4 1 2 3 5 1	4 2 1 3 4 1 2 3 5 1	4 2 1 3 4 1 2 3 5 1	✓
✓	`Node* ll = vectorToLinkedList(values);` `print(ll);`	10 4 2 1 3 4 1 2 3 5 1	4 2 1 3 4 1 2 3 5 1	4 2 1 3 4 1 2 3 5 1	✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
vector<int> linkedListToVector(Node* head)
{
    // Your code here
    vector<int> ret;
    while (head) {
        ret.push_back(head->value);
        head = head->next;
    }
    return ret;
}

Node* vectorToLinkedList(const vector<int> &values, int i) {
    if (i >= int(values.size())) return NULL;
    Node* new_node = new Node();
    new_node->value = values[i];
    new_node->next = vectorToLinkedList(values, i+1);
    return new_node;
}

Node* vectorToLinkedList(vector<int> values)
{
    return vectorToLinkedList(values, 0);
}
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 4

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một `struct Node` biểu diễn một $node$ của 1 danh sách liên kết đơn như sau:

```
struct Node
{
    int value;
    Node* next;
};
```

Viết hàm: `Node* extractNodes(Node* head, int threshold)` tạo và trả về một danh sách liên kết mới từ các $node$ trong danh sách liên kết bắt đầu bằng $head$, các $node$ trong danh sách liên kết mới đều có giá trị nhỏ hơn $threshold$. Hàm không làm thay đổi danh sách liên kết cũ.

Gợi ý:

1. Duyệt lần lượt từng phần tử và lấy ra từng phần tử, thêm vào cuối của danh sách liên kết mới.
2. Nên lưu lại tail để thao tác nhanh hơn.

For example:

Test	Input	Result
`Node* otherHead = extractNodes(head, 3);` `print(otherHead);`	4 3 2 1 4	2 1

Answer: (penalty regime: 0 %)

Reset answer

```
Node* extractNodes(Node* head, int threshold)
{
    // Your code here
    Node *newhead = NULL;
    Node *p = head;
    Node *q;
    while (p) {
        if (p->value < threshold) {
            if (!newhead) {
                newhead = new Node;
                newhead->value = p->value;
                q = newhead;
            } else {
                Node *r = new Node;
                r->value = p->value;
                q->next = r;
                q = r;
            }
        }
        p = p->next;
    }
    return newhead; // Change this line
}
```

	Test	Input	Expected	Got	
✓	`Node* otherHead = extractNodes(head, 3);` `print(otherHead);`	4 3 2 1 4	2 1	2 1	✓

	Test	Input	Expected	Got	
✓	Node* otherHead = extractNodes(head, 0); print(otherHead);	4 3 2 1 4			✓
✓	Node* otherHead = extractNodes(head, 5); print(otherHead);	4 3 2 1 4	3 2 1 4	3 2 1 4	✓

✓	Node* headA = extractNodes(head, 5);
	Node* headB = extractNodes(headA, 3);
	print(head); cout << endl;
	print(headA); cout << endl;
	print(headB); cout << endl;
		10
		2 5 8 4 6 7 9 1 3 5
			2 5 8 4 6 7 9 1 3 5
			2 4 1 3
			2 1
				2 5 8 4 6 7 9 1 3 5
				2 4 1 3
				2 1
					✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
Node* extractNodes(Node* head, int threshold)
{
    if (head == NULL) return NULL;
    if (head->value >= threshold)
        return extractNodes(head->next, threshold);
    // copy a new Node, since we are not allowed to change the original linked-list
    Node* new_node = new Node();
    new_node->value = head->value;
    new_node->next = extractNodes(head->next, threshold);
    return new_node;
}
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **5**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một `struct Node` biểu diễn một *node* của 1 danh sách liên kết đơn như sau:

```
struct Node
{
    int value;
    Node* next;
};
```

Viết hàm `Node* concat(vector<Node*> heads);` nhận đầu vào là một vector chứa các *node* đầu tiên của 1 tập các danh sách liên kết, nối các danh sách liên kết đầu vào thành 1 danh sách liên kết mới và trả về *node* đầu tiên của danh sách liên kết mới đó.

Gợi ý:

1. Duyệt từng danh sách liên kết.
2. Với mỗi danh sách liên kết, duyệt đến node cuối cùng.
3. Gán next của node cuối cùng bằng head của danh sách tiếp theo.

For example:

Test	Result

`int a[] = {1, 2, 3};`
`int b[] = {4, 5, 6};`
`Node* headA = createLinkedList(a, 3);`
`Node* headB = createLinkedList(b, 3);`
`vector<Node*> heads;`
`heads.push_back(headA);`
`heads.push_back(headB);`
`Node* headCombined = concat(heads);`
`print(headCombined);`
	`1 2 3 4 5 6`

Answer: (penalty regime: 0 %)

Reset answer

```
Node* concat(vector<Node*> heads)
{
    // Your code here
    Node *bighead = NULL;
    Node *bigtail;
    for (Node *head : heads) {
        Node *p = head;
        if (!p) {
            continue;
        }
        if (!bighead) {
            bighead = new Node;
            bighead->value = p->value;
            bigtail = bighead;
            p = p->next;
        }
        while (p) {
            Node *q = new Node;
            q->value = p->value;
            bigtail->next = q;
            bigtail = q;
            p = p->next;
        }
    }
    return bighead; // Change this line
}
```

	Test	Expected	Got	

✓	int a[] = {1, 2, 3};
	int b[] = {4, 5, 6};
	Node* headA = createLinkedList(a, 3);
	Node* headB = createLinkedList(b, 3);
	vector<Node*> heads;
	heads.push_back(headA);
	heads.push_back(headB);
	Node* headCombined = concat(heads);
	print(headCombined);
		1 2 3 4 5 6	1 2 3 4 5 6	✓

✓	int a[] = {1, 2, 3};
	int b[] = {};
	int c[] = {4, 5, 6};
	Node* headA = createLinkedList(a, 3);
	Node* headB = createLinkedList(b, 0);
	Node* headC = createLinkedList(c, 3);
	vector<Node*> heads;
	heads.push_back(headA);
	heads.push_back(headB);
	heads.push_back(headC);
	Node* headCombined = concat(heads);
	print(headCombined);
	cout << endl;
	print(headC);
		1 2 3 4 5 6
		4 5 6
			1 2 3 4 5 6
			4 5 6
				✓

✓	int a[] = {};
	int b[] = {1, 2, 3};
	int c[] = {};
	Node* headA = createLinkedList(a, 0);
	Node* headB = createLinkedList(b, 3);
	Node* headC = createLinkedList(c, 0);
	vector<Node*> heads;
	heads.push_back(headA);
	heads.push_back(headB);
	heads.push_back(headC);
	Node* headCombined = concat(heads);
	print(headCombined);
		1 2 3	1 2 3	✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
Node* concat(vector<Node*> heads)
{
    Node* head = NULL;
    for (int i=0, j=-1; i < int(heads.size()); ++i) {
        if (heads[i] == NULL) continue;
        if (j >= 0 && heads[j] != NULL) {
            while (heads[j]->next != NULL) heads[j]=heads[j]->next;
            heads[j]->next = heads[i];
        }
        if (head == NULL) head = heads[i];
        j=i;
    }
    return head;
}
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 6

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một `struct Node` biểu diễn một *node* của 1 danh sách liên kết đơn như sau:

```
struct Node
{
    int value;
    Node* next;
};
```

Viết hàm `void printReverse(Node* head);` in các giá trị của danh sách liên kết theo chiều ngược.

Gợi ý:

1. Tham khảo video đệ quy với danh sách liên kết.

For example:

Test	Input	Result
printReverse(head);	4 1 2 3 4	4 3 2 1

Answer: (penalty regime: 0 %)

Reset answer

```
void printReverse(Node* head)
{
    if (!head) {
        return;
    }
    if (head->next) {
        printReverse(head->next);
    }
    cout << head->value << ' ';
}
```

	Test	Input	Expected	Got	
✓	printReverse(head);	4 1 2 3 4	4 3 2 1	4 3 2 1	✓
✓	printReverse(head);	1 1	1	1	✓

	Test	Input	Expected	Got	
✓	printReverse(head);	10 2 1 5 7 8 1 5 4 3 2	2 3 4 5 1 8 7 5 1 2	2 3 4 5 1 8 7 5 1 2	✓
✓	printReverse(head);	0			✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
void printReverse(Node* head)
{
    // Your code here
    if (head == NULL) return;
    printReverse(head->next);
    cout << head->value << " ";
}
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 7

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một `struct Node` biểu diễn một $node$ của 1 danh sách liên kết đơn như sau:

```
struct Node
{
    int value;
    Node *next;
};
```

Biết $head$ là con trỏ trỏ tới một danh sách liên kết đã được sắp xếp tăng dần theo giá trị các $node$, viết các hàm sau:

- `Node* deleteDuplicates(Node* head);` xoá các $node$ khỏi danh sách liên kết sao cho các giá trị xuất hiện nhiều nhất một lần. Hàm trả về con trỏ trỏ tới vị trí đầu tiên của danh sách liên kết.
- `Node* insert(Node* head, int value);` chèn một $node$ có giá trị $value$ vào danh sách liên kết sao cho danh sách liên kết vẫn được sắp xếp tăng dần. Hàm trả về con trỏ trỏ tới vị trí đầu tiên của danh sách liên kết.

Gợi ý:

1. Để xoá trùng: tại mỗi node, nếu giá trị của nó bằng giá trị của next thì xoá node next bằng cách gán p->next = p->next->next.
2. Để chèn: tham khảo video hướng dẫn đệ quy và danh sách liên kết.

For example:

Test	Input	Result
`head = deleteDuplicates(head);` `print(head);`	6 2 2 3 3 3 4	2 3 4
`head = insert(head, 3);` `print(head);`	6 2 2 3 3 3 4	2 2 3 3 3 3 4

Answer: (penalty regime: 0 %)

Reset answer

```
Node* deleteDuplicates(Node* head)
{
    // Your code here
    if (!head || !head->next) {
        return head;
    }
    Node *p = head;
    Node *q = head->next;
    Node *newhead = new Node;
    newhead->value = head->value;
    Node *newtail = newhead;
    while (q) {
        if (q->value != p->value) {
            Node *r = new Node;
            r->value = q->value;
            newtail->next = r;
            newtail = r;
        }
        p = p->next;
        q = q->next;
    }
    return newhead;
}

Node* insert(Node* head, int value)
{
    // Your code here
    Node *p = new Node;
    if (value <= head->value) {
```

```
    if (value <= head->value) {
        p->value = value;
        p->next = head;
        head = p;
    } else {
        p = head;
        while (p->next && p->next->value < value) {
            p = p->next;
        }
        Node *q = new Node;
        q->value = value;
        q->next = p->next;
        p->next = q;
    }
    return head;
}
```

	Test	Input	Expected	Got	
✓	head = deleteDuplicates(head); print(head);	6 2 2 3 3 3 4	2 3 4	2 3 4	✓
✓	head = insert(head, 3); print(head);	6 2 2 3 3 3 4	2 2 3 3 3 3 4	2 2 3 3 3 3 4	✓
✓	head = deleteDuplicates(head); print(head);	4 1 2 3 4	1 2 3 4	1 2 3 4	✓
✓	head = deleteDuplicates(head); print(head);	6 4 4 4 4 4 4	4	4	✓
✓	head = deleteDuplicates(head); print(head);	6 1 1 1 2 2 2	1 2	1 2	✓
✓	head = deleteDuplicates(head); print(head);	0			✓
✓	head = insert(head, 1); print(head);	3 2 3 4	1 2 3 4	1 2 3 4	✓
✓	head = insert(head, 4); print(head);	4 1 2 3 3	1 2 3 3 4	1 2 3 3 4	✓
✓	head = insert(head, 2); print(head);	4 1 1 3 5	1 1 2 3 5	1 1 2 3 5	✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```
Node* deleteDuplicates(Node* head)
{
    // Your code here
    if (head == NULL) return head;
    for (Node* it = head; it != NULL; it=it->next) {
        while (it->next && it->next->value == it->value) {
            Node* tmp = it->next;
            it->next = tmp->next;
            delete tmp;
        }
    }
    return head;
}

Node* insert(Node* head, int value)
{
    // Your code here
    if (head != NULL && head->value < value) {
        head->next = insert(head->next, value);
        return head;
```

```
    }
    Node* new_node = new Node();
    new_node->value = value;

    if (head == NULL) return new_node;
    new_node->next = head;
    return new_node;
}
```

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 8

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một danh sách liên kết như sau:

$a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_n \rightarrow b_1 \rightarrow b_2 \rightarrow \ldots \rightarrow b_n$

Không sử dụng thêm mảng phụ, hãy viết hàm `Node* convert(Node* head);` để chuyển đổi danh sách liên kết trên thành:

$a_1 \rightarrow b_1 \rightarrow a_2 \rightarrow b_2 \rightarrow \ldots \rightarrow a_n \rightarrow b_n$

Gợi ý:

Cách tìm phần tử b1

1. Cách chậm: đếm số phần tử, duyệt n/2 lần sẽ gặp b1.
2. Cách nhanh: dùng 2 con trỏ, pSlow và pFast cùng xuất phát từ head. Mỗi lần lặp: pSlow = pSlow->next, pFast = pFast->next->next. Như vậy, khi pFast đến cuối thì pSlow sẽ nằm ở giữa.

For example:

Input	Result
6 1 3 5 2 4 6	1 2 3 4 5 6

Answer: (penalty regime: 0 %)

Reset answer

```
Node* convert(Node* head)
{
    // Your code here
    if (!head) {
        return head;
    }
    Node *pSlow = head;
    Node *pFast = head;
    while (pFast) {
        pSlow = pSlow->next;
        pFast = pFast->next->next;
    }
    Node *pa = head->next;
    Node *pb = pSlow;
    Node *p = head;
    for (int i = 1; pb; ++i) {
        if (i & 1) {
            p->next = pb;
            p = pb;
            pb = pb->next;
        } else {
            p->next = pa;
            p = pa;
            pa = pa->next;
        }
    }
    return head;
}
```

	Input	Expected	Got	
✓	6 1 3 5 2 4 6	1 2 3 4 5 6	1 2 3 4 5 6	✓

Passed all tests! ✓

▾ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

```cpp
Node* convert(Node* head)
{
    // Your code here
    if (head == NULL) return NULL;
    Node* fast = head, *slow = head;
    Node* prev_slow = NULL;
    while (fast != NULL) {
        prev_slow = slow;
        fast = fast->next->next;
        slow = slow->next;
    }

    Node* a = head, *b = slow;
    prev_slow->next = NULL; // cut the two arrays
    while (b != NULL) {
        Node* a2 = a->next;
        a->next = b;
        b = b->next;
        a = a->next;
        a->next = a2;
        a = a->next;
    }
    return head;
}
```

Đúng

Marks for this submission: 10,00/10,00.

Trở lại Khoá học