Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Ba, 22 tháng 4 2025, 11:06 AM
Kết thúc lúc	Thứ Ba, 22 tháng 4 2025, 11:51 AM
Thời gian thực hiện	45 phút 1 giây
Điểm	33,33/50,00
Điểm	**6,67** trên 10,00 (**66,67**%)

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;
};
```

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)
{
    Node *res=nullptr;
    Node *temp=head;
    Node *tail=nullptr;
    while(temp!=nullptr){
        if(temp->value<threshold){
            Node *newNode = new Node(*temp);
            newNode->next=nullptr;
            if(res==nullptr){
                res=newNode;
            }
            else{
                tail->next=newNode;
            }
            tail=newNode;
        }
        temp=temp->next;
    }
    return res; // Change this line
}
```

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

Sai

Đạt điểm 0,00 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)
{
    Node *
    return NULL; // Change this line
}
```

Syntax Error(s)

```
__tester__.cpp: In function ‘Node* concat(std::vector<Node*>)’:
__tester__.cpp:51:5: error: expected unqualified-id before ‘return’
   51 |     return NULL; // Change this line
      |     ^~~~~~
__tester__.cpp:52:1: error: no return statement in function returning non-void [-Werror=return-type]
   52 | }
      | ^
cc1plus: all warnings being treated as errors
```

▾ 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;
}
```

Sai

Marks for this submission: 0,00/10,00.

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 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)
{
    Node *prev=nullptr;
    Node *current=head;
    Node *next=new Node;
    while(current!=nullptr){
        next=current->next;
        current->next=prev;
        prev=current;
        current=next;
    }
    Node *temp=prev;
    while(temp!=nullptr){
        cout<<temp->value<<" ";
        temp=temp->next;
    }
}
```

	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	✓
✓	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)

```
1  void printReverse(Node* head)
2  {
3      // Your code here
4      if (head == NULL) return;
5      printReverse(head->next);
6      cout << head->value << " ";
7  }
```

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;
};
```

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)
{
    Node *temp=head;
    while(temp!=nullptr && temp->next!=nullptr){
        if(temp->value==temp->next->value){
            Node *dup=temp->next;
            temp->next=temp->next->next;
            delete dup;
        }
        else{
            temp=temp->next;
        }
    }
    return head;
}

Node* insert(Node* head, int x)
{
    Node *newNode=new Node{x, nullptr};
    if(head==nullptr || x<head->value){
        newNode->next=head;
        return newNode;
```

```
    }
    Node *temp=head;
    while(temp->next && temp->next->value<x){
        temp=temp->next;
    }
    newNode->next=temp->next;
    temp->next=newNode;
    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 **5**

Sai

Đạt điểm 3,33 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)
{

    Node *slow=head;
    Node *fast=head;
    while(fast!=nullptr && fast->next!=nullptr){
        slow=slow->next;
        fast=fast->next->next;
    }
    Node *first=head;
    Node *second=slow;
    while(second!=nullptr && second->next!=nullptr){
        Node *firstNext=first->next;
        Node *secondNext=second->next;

        first->next=second;
        second->next=firstNext;
        first=firstNext;
        second=secondNext;

    }
    return head;
}
```

	Input	Expected	Got	
✕	6 1 3 5 2 4 6	1 2 3 4 5 6	1 2 3 4 5 ...snip... 5 2 3 4 5 2 3	✕

	Input	Expected	Got	
			4 5 2 3 4 ***Time limit exceeded***	

Testing was aborted due to error.

Show differences

SHOW/HIDE QUESTION AUTHOR'S SOLUTION (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;
```

```
    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;
}
```

Sai

Marks for this submission: 0,00/10,00.

Trở lại Khoá học