Bắt đầu vào lúc	Thứ Hai, 1 tháng 4 2024, 7:50 PM
Trạng thái	Đã xong
Kết thúc lúc	Thứ Hai, 1 tháng 4 2024, 7:59 PM
Thời gian thực hiện	9 phút 26 giây
Điểm	17,00/18,00
Điểm	**9,44** trên 10,00 (**94,44**%)

Câu hỏi 1

Đúng

Đạt điểm 1,00 trên 1,00

Chọn câu đúng khi so sánh Singly linked list và Array:

a. Kích thước của Array phải được cho trước, Linked List thì không cần.

b. Tất cả đều đúng ✓

c. Chèn và xóa trong Linked List dễ hơn so với Array

d. Array cho phép truy xuất đến các giá trị trong mảng dễ dàng hơn Linked List

e. Truy cập ngẫu nhiên không được phép trong Linked Lists

Your answer is correct.

The correct answer is:
Tất cả đều đúng

Câu hỏi 2

Đúng

Đạt điểm 1,00 trên 1,00

Cho P là một danh sách liên kết đơn. Gọi Q là con trỏ tới một node trung gian x trong danh sách. Độ phức tạp thời gian trong trường hợp xấu nhất của thuật toán tốt nhất để xóa node x khỏi danh sách là:

a. $O(n^2)$

b. $O(\log_2 n)$

c. $O(n)$

d. $O(1)$ ✓

Your answer is correct.

A simple solution is to traverse the linked list until you find the node you want to delete. But this solution requires pointer to the head node which contradicts the problem statement. Fast solution is to copy the data from the next node to the node to be deleted and delete the next node. Something like following.

```
// Find next node using next pointer
struct node *temp  = node_ptr->next;

// Copy data of next node to this node
node_ptr->data  = temp->data;

// Unlink next node
node_ptr->next  = temp->next;

// Delete next node
free(temp);
```

Time complexity of this approach is O(1) Refer this for implementation. Note that this approach doesn't work when node to deleted is last node. Since the question says intermediate node, we can use this approach.

The correct answer is:
O(1)

Câu hỏi 3

Đúng

Đạt điểm 3,00 trên 3,00

Tạo danh sách liên kết đơn lưu trữ các số nguyên. Các số được lần lượt thêm vào cuối, vào đầu danh sách, hoặc sau một số biết trước. Input gồm nhiều dòng, mỗi dòng sẽ có cấu trúc ở một trong 4 dạng sau:

- Dạng 0: Dòng bắt đầu bằng con số 0, theo sau là một số nguyên, chương trình **thêm con số này vào đầu** danh sách.

- Dạng 1: Dòng bắt đầu bằng con số 1, theo sau là một số nguyên, chương trình **thêm con số này vào cuối** danh sách.

- Dạng 2: Dòng bắt đầu bằng con số 2, theo sau là 2 số nguyên **X và Y**, chương trình **thêm con số Y vào sau con số X (nếu X có trong danh sách)**.

Hãy cho biết kết quả danh sách liên kết được in từ đầu đến cuối mảng.

Lưu ý: Các giá trị trong danh sách ghi cách nhau một khoảng trắng, và không ghi khoảng trắng ở đầu và cuối danh sách.

Input 1:

0 6

2 4 5

0 4

1 2

2 6 3

Output 1: 4 6 3 2 ✓

Input 2:

0 8

0 19

1 5

1 2

0 20

1 9

2 8 1

2 20 4

1 5

2 15 25

0 10

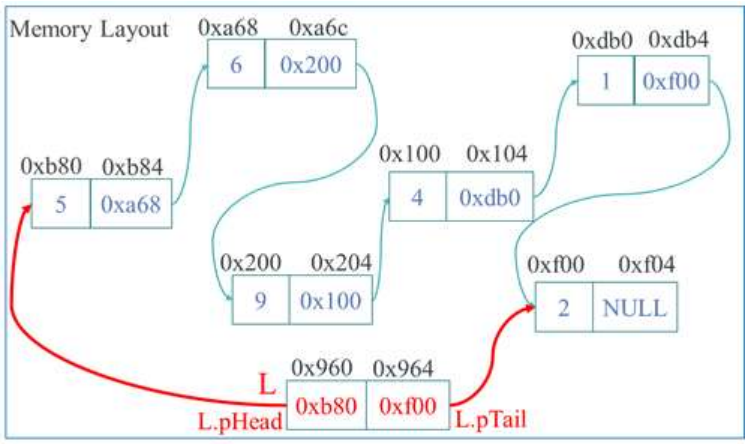
Output 2: 10 20 4 19 8 1 5 2 9 5 ✓

Câu hỏi 4

Sai

Đạt điểm 0,00 trên 1,00

Cho danh sách liên kết đơn L lưu các giá trị sau: {5, 6, 9, 4, 1, 2}. Giả định rằng các phần tử có sự liên kết như hình bên dưới.



Cấu trúc của node và danh sách như sau:

```
struct NODE {
    int info;
    NODE* pNext;
};
struct LIST {
    NODE* pHead;
    NODE* pTail;
};
```

Hãy viết lệnh để truy xuất đến giá trị pNext = 0x100.

Chú ý: không sử dụng toán tử *, khoảng trắng, dấu ngoặc tròn trong câu trả lời.

Answer: pHead->pNext->pNext->pNext->info ✕

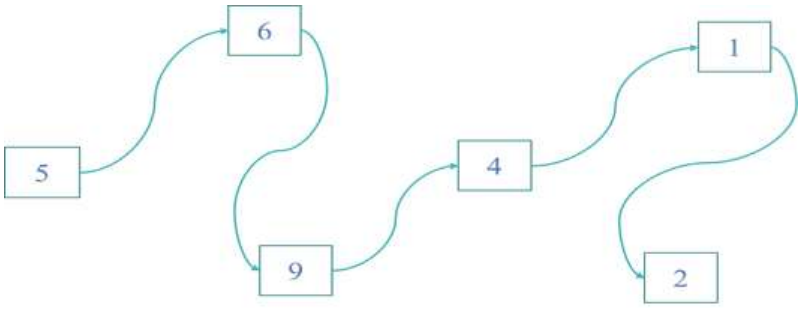
The correct answer is: L.pHead->pNext->pNext->pNext

Câu hỏi 5

Đúng

Đạt điểm 1,00 trên 1,00

Điền nội dung còn thiếu để đoạn code sau chạy được và ra kết quả danh sách liên kết đơn như hình minh họa (5 là phần tử đầu danh sách, 2 là phẩn tử cuối danh sách).



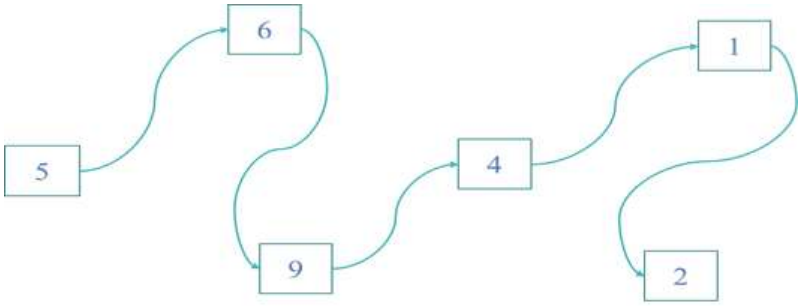

```
struct NODE {
    int info;
    NODE* pNext;
};
struct LIST {
    NODE* pHead;
    NODE* pTail;
};
void AddTail(LIST &, const int &);
void AddHead(LIST &, const int &);

int main() {
    LIST L;
    CreateEmptyList(L);
    AddHead(L, 9);
    AddHead(L, 6);
    AddTail(L, 4);
    AddHead(L, 5);
    AddTail(L, 1);
    AddTail(L, 2);

    return 0;
}
```

AddTail | AddHead

Your answer is correct.

The correct answer is:

Điền nội dung còn thiếu để đoạn code sau chạy được và ra kết quả danh sách liên kết đơn như hình minh họa (5 là phần tử đầu danh sách, 2 là phần tử cuối danh sách).




```
struct NODE {
    int info;
    NODE* pNext;
};
struct LIST {
    NODE* pHead;
    NODE* pTail;
};
void AddTail(LIST &, const int &);
void AddHead(LIST &, const int &);

int main() {
    LIST L;
    CreateEmptyList(L);
    AddHead(L, 9);
    [AddHead](L, 6);
    [AddTail](L, 4);
    [AddHead](L, 5);
    [AddTail](L, 1);
    [AddTail](L, 2);

    return 0;
}
```

Câu hỏi 6

Đúng

Đạt điểm 7,00 trên 7,00

Hãy điền vào vị trí trống bên dưới để được hàm AddHead dùng thêm X vào cuối danh sách liên kết đơn:
Lưu ý: Những vị trí tự điền kết quả thì câu trả lời KHÔNG có khoảng trắng.

```
struct NODE {
    int info;
    NODE*  pNext;
};
struct LIST {
    NODE* pHead;
    NODE* pTail;
};
NODE*  CreateNode(const int &x) {
    NODE* p  = new  NODE  ;
    p->info  = x;
    p->pNext = NULL;
    return p;
}
void AddHead(  LIST &L  , NODE* p) {
    if (L.pHead == NULL) {
        L.pHead =  p  ;
        L.pTail = L.pHead;
    }
    else {
        p->pNext  = L.pHead;
        L.pHead  = p;
   }
}
```

Câu hỏi 7

Đúng

Đạt điểm 1,00 trên 1,00

Điền vào các vị trí khuyết trong hàm sum để tính tổng tất cả các node trong danh sách. Biết rằng đầu vào của hàm là con trỏ trỏ tới phần tử đầu của danh sách.

```
struct NODE {
    int info;
    NODE* pNext;
};
void sum(NODE* p, int &s ✓ ) {
    s=0;

    while ( p ✓ ) {
        s += p->info ✓ ;
        p=p->pNext ✓ ;
    }
}
```

Your answer is correct.

The correct answer is:

Điền vào các vị trí khuyết trong hàm sum để tính tổng tất cả các node trong danh sách. Biết rằng đầu vào của hàm là con trỏ trỏ tới phần tử đầu của danh sách.

```
struct NODE {
    int info;
    NODE* pNext;
};
void sum(NODE* p, [int &s]) {
    s=0;

    while ([p]) {
        s += [p->info];
        [p=p->pNext];
    }
}
```

Câu hỏi 8

Đúng

Đạt điểm 1,00 trên 1,00

Điền vào chỗ khuyết trong hàm **RemoveHead** để xóa node đầu danh sách L:

```
struct NODE {
    int info;
    NODE *pNext;
};
struct LIST {
    NODE *pHead;
    NODE *pTail;
};
void RemoveHead(LIST &L) {
    NODE *p;
    if ( L.pHead != NULL ✓ ) {
        p = L.pHead;
        L.pHead ✓ = p->pNext;
        delete p;
        if ( L.pHead == NULL ✓ )
            L.pTail ✓ = NULL;
    }
}
```

Your answer is correct.

The correct answer is:
Điền vào chỗ khuyết trong hàm **RemoveHead** để xóa node đầu danh sách L:

```
struct NODE {
    int info;
    NODE *pNext;
};
struct LIST {
    NODE *pHead;
    NODE *pTail;
};
void RemoveHead(LIST &L) {
    NODE *p;
    if ([L.pHead != NULL]) {
        p = L.pHead;
        [L.pHead] = p->pNext;
        delete p;
        if ([L.pHead == NULL])
            [L.pTail] = NULL;
    }
}
```

Câu hỏi 9

Đúng

Đạt điểm 1,00 trên 1,00

Điền vào các vị trí khuyết trong hàm **print** để in tất cả các node của danh sách ra màn hình. Nếu danh sách rỗng thì in ra **"Empty List!"**. Biết tham số đầu vào **p là con trỏ trỏ đến phần tử đầu** tiên trong danh sách.

```
void print(NODE* p) {
    if ( p==NULL ✓ ) cout << "Empty List!";
    while (p) {
        cout << p->info << " ";
        p=p->pNext ✓ ;
    }
}
```

p=head->pNext | p!=NULL | p=NULL | p=p.pNext

Your answer is correct.

The correct answer is:

Điền vào các vị trí khuyết trong hàm **print** để in tất cả các node của danh sách ra màn hình. Nếu danh sách rỗng thì in ra **"Empty List!"**. Biết tham số đầu vào **p là con trỏ trỏ đến phần tử đầu** tiên trong danh sách.

```
void print(NODE* p) {
    if ([p==NULL]) cout << "Empty List!";
    while (p) {
        cout << p->info << " ";
        [p=p->pNext];
    }
}
```

Câu hỏi 10

Đúng

Đạt điểm 1,00 trên 1,00

Điền vào các vị trí khuyết trong hàm count_even để đếm số node có giá trị chẵn trong danh sách.

```
int count_even(const LIST [ & ] ✓ L) {
    int count=0;
    NODE *p=[ L.pHead ] ✓ ;
    while (p) {
        if ( [ p->info%2==0 ] ✓ )
            count++;
        p=[ p->pNext ] ✓ ;
    }
    return [ count ] ✓ ;
}
```

Your answer is correct.

The correct answer is:

Điền vào các vị trí khuyết trong hàm count_even để đếm số node có giá trị chẵn trong danh sách.

```
int count_even(const LIST [&]L) {
    int count=0;
    NODE *p=[L.pHead];
    while (p) {
        if ([p->info%2==0])
            count++;
        p=[p->pNext];
    }
    return [count];
}
```

◄ Quiz 5: Heap sort (Đã mở lại và gia hạn thêm thời gian)

Chuyển tới...

Quiz 7: stack -queue ►