No.
Date

單元2：抽象化

* class of objects (call instances)
  Attributes : data members
  Behaviors : methods

* Three characteristics
  - Encapsulation : hides inner details (封裝)
  - Inheritance : reused (繼承)
  - Polymorphism (多型)

* Operation Contracts 運算合約
  — A module's operation contract specifies its
  - Purpose
  - Assumptions
  - Input
  - Output

* Key Issues in Programming

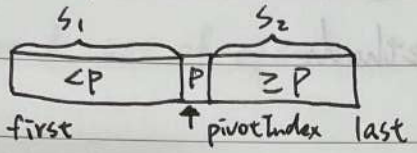
1. Modularity
2. Style
3. Modifiability
4. Ease of Use
5. Fail-safe programming
6. Debugging
7. Testing

單元 1：遞迴

No.
Date : :

Finding the k^{th} Smallest Item in an Array

* pivot item 樞紐



Linear Recursion

* Test for base cases 檢查終止條件
* Recur once 只撰一遞迴

Binary Recursion

* Occurs whenever there are two recursive calls for each non-base case

Summary

1. 遞迴定義
2. 問題簡化
3. 終止條件 (base case)
4. 保證終止

No.
Date : :

* Inheritance in C++

父類別 : base class

子類別 : derived class

- An instance of a derived class can invoke public methods of the base class.

* C++ Namespaces

- Creating a namespace

自訂命名空間

```
namespace smallNamespace
{
    int count = 0;
    void abc();
}
```

- Using a namespace

使用命名空間

```
using namespace smallNamespace;
count += 1;
abc();
```

* using namespace std;

No.
Date : :

* Abstract Data Type (ADT)

- An ADT is composed of
  - A collection of data
  - A set of operations on that data

- Specifications of an ADT indicate 描述
  - What the ADT operations, not how to implement them

- Implementation of an ADT 實作
  - Includes choosing a particular data structure

* Constructors

- Create and initialize new instances of a class
- Have the same name as the class
- Have no return type, not even void

* Destructors

- Destroys an instance of an object when the object's lifetime ends

單元3：鏈接串列

No.
Date : :

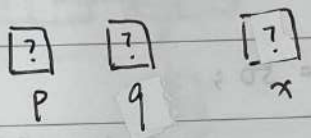
Pointer(指標) int *p; 指標 = 門牌

p = &x; &x = 房子x的門牌

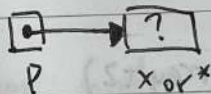
p = new int; (Dynamic allocation)

delete p;
p = NULL; ↑ 刪除後記得設為NULL，不然會被其他程式使用

(a) int *p, *q;
int ;



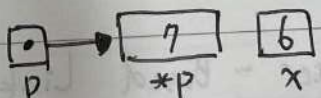
(b) p = &x;
(b要先做，c才能做)



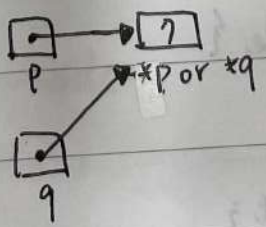
(c) *p = 6;

(d) p = new int;

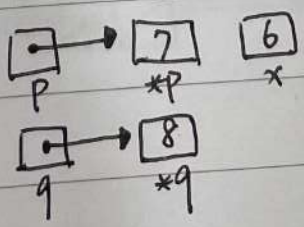
(e) *p = 7;



(f) q = p;



(g) q = new int;
*q = 8;

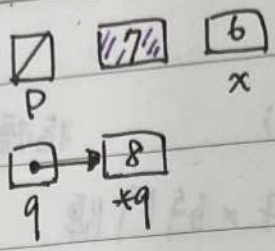


No.

Date : memory leak

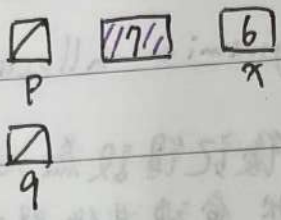
(h) p = NULL;

(X) 錯誤作法



(i) delete q;
q = NULL;

(O) 正確作法



動態陣列

```
int arraySize = 50;
double *anArray = new double [arraySize];
```

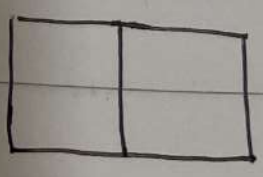
anArray[2] ≡ *(anArray+2)　　陣列名稱三指標

配置更大空間!

```
double *oldArray = anArray;
anArray = new double [3 * arraySize];
```

Pointer - Based Linked List

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



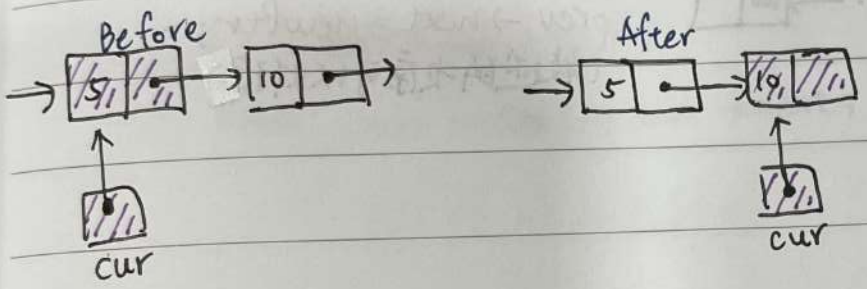
No.
Date : :

* A node is dynamically allocated

```
Node *p;
p = new Node;
```

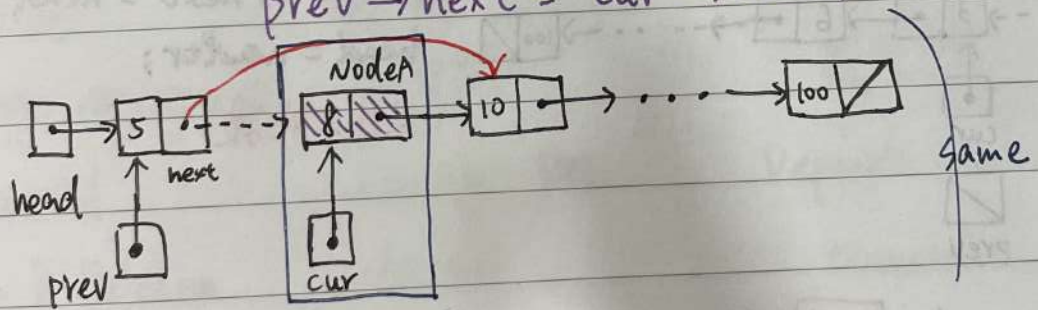
* `for (Node * cur = head; cur != NULL; cur = cur->next)`

```
cout << cur->item << endl;
```



* Deleting an interior node

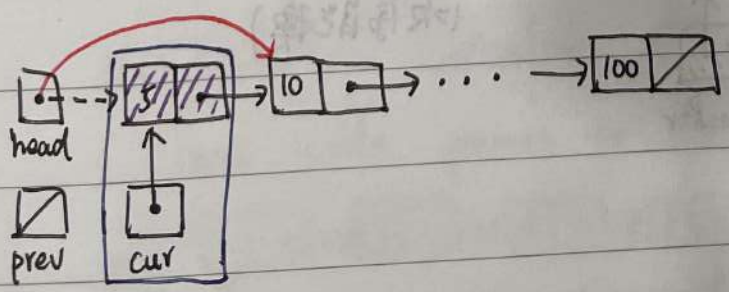
```
prev->next = cur->next;
```




```
prev->next = prev->next->next;
```

* Deleting the first node

```
head = cur->next;
```



No.
Date : :

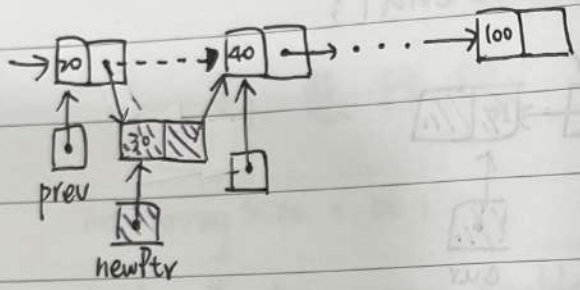
* Return deleted node to system

```
cur -> next = NULL;
delete cur;
cur = NULL;
```

不能寫成→

```
cur -> next = NULL;
cur = NULL;
delete cur;
```

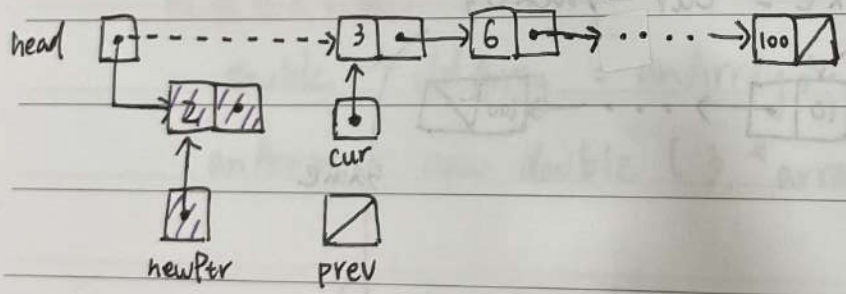
* To insert a node between two nodes




```
newPtr -> next = cur;
prev -> next = newPtr;
```

(敘述的次序可以對調)

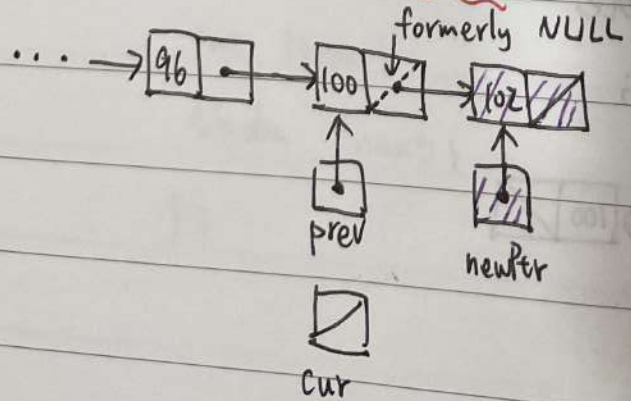
* To insert a node at the beginning of a linked list




```
newPtr -> next = head;
head = newPtr;
```

(次序不能換)

* Inserting at the end of a linked list is not a special case if cur is NULL




```
newPtr -> next = cur;   → NULL
prev -> next = newPtr;
```

(次序能換)

No.
Date : :

* Delete a Node from a sorted linked list

```
Node *prev, *cur;
if (head != NULL) {
    for (prev = NULL, cur = head;
         (cur != NULL) && (newValue > cur->item);
         prev = cur, cur = cur->next);
    if (prev == NULL)
        head = cur->next;
    else prev->next = cur->next;
```

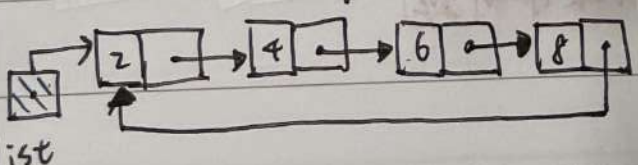
	Array	Linked list
Size	固定	彈性
Storage requirements	省空間	花空間
Retrieval	Constant (快)	Depends on i (慢)
Insertion & Deletion	shifting	traversal

Passing a Linked List to a Method

傳址& : 更改指標內容

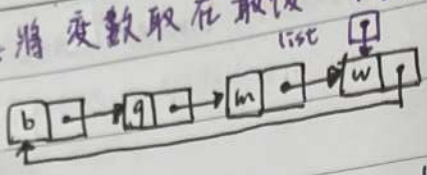
Variations : Circular Linked lists

last node points to the first node



No.
Date : :

*将变数取在最後一个节点



Variations: Dummy Head Node

* Eliminates the special case

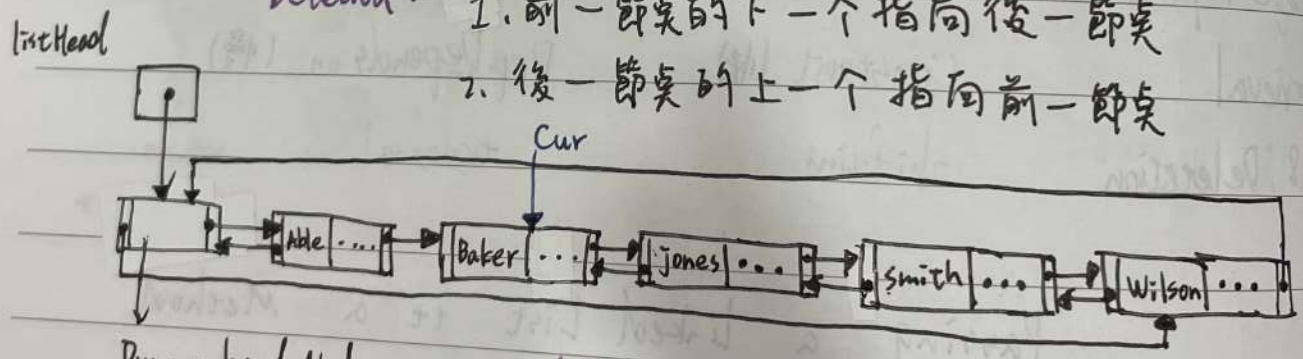
Deletion: 1. 走訪至欲刪除的節点
2. 改变指標
3. 釋放空間

Insertion: 1. 走訪至欲新增的前一个位置
2. 改变指標

Variations: Doubly 双向 Linked Lists

*在 next 之外多一个指標指向前面

Deletion: 1. 前一節点的下一个指向後一節点
2. 後一節点的上一个指向前一節点



To delete the node to which cur points

(cur→precede)→next = cur→next;

(cur→next)→precede = cur→precede;

(敘述的次序可換)

單元4：以遞迴解題

* Defining Languages
  - A language
  - A grammar

* Algebraic Expression
  - Infix expressions 中序運算式
    - An operator (運算子) appears between its operands (運算元)

      Example : a + b

  - Prefix expressions
    - An operator appears before its operands 前序運算式

      Example : + ab
  - Postfix expressions
    - An operator appears after its operands 後序運算式

      Example : ab +
  - Fully parenthesized infix expressions 完整括號

No.
Date : :

Insert:

To insert a new node pointed to by newPtr
before the node pointed to by cur

```
newPtr → next = cur;
newPtr → precede = cur → precede;
cur → precede = newPtr;
newPtr → precede → next = newPtr;
```

(次序不能換!)