



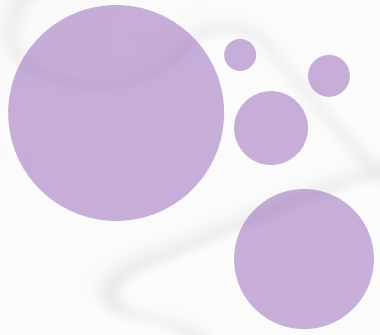
2018

**程式設計
沒有加強班**

程式設計與實習(二)

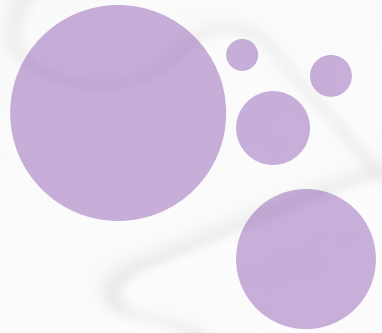
BY 孫茂勛

筆電壞掉了所以只有微軟正黑體QQ
Email:JOHN85051232@GMAIL.COM



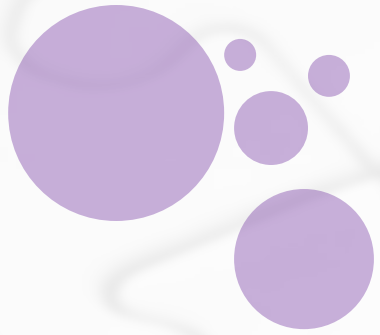
事情宣布

🔵 5/8(二)大一同學(學號A10655__)記得要來進行APCS試測，非大一同學不用參加~



複習一下你可能已經忘光的東西





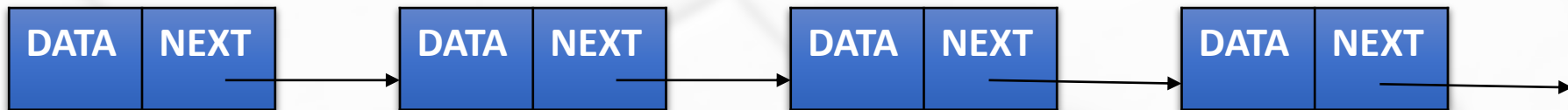
Linked list

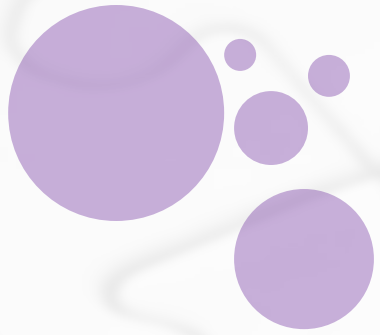
鏈結串列(Linked list)：

- ❖ 資料結構的一種

- ❖ struct + pointer

- ❖ 每一個節點(node)都包含指向下一個節點的指標





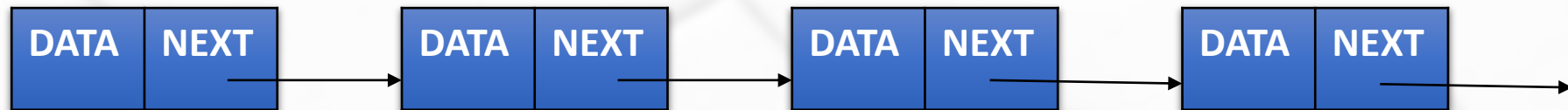
Linked list

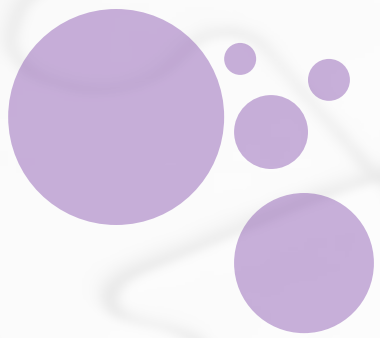
基本操作：

🔷移動(走訪)

🔷新增

🔷刪除





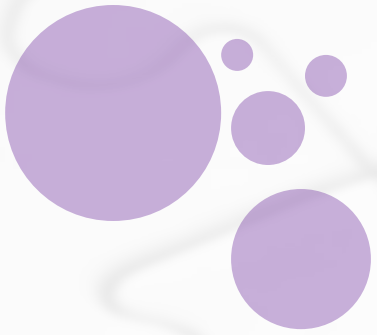
Linked list

先建好結構

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int num;
    node *next;
};
```

```
int main( )
{
    node *n1 = new node;
    node *n2 = new node;
    node *n3 = new node;
    n1->num = 10;
    n1->next = n2;
    n2->num = 20;
    n2->next = n3;
    n3->num = 30;
    n3->next = NULL;

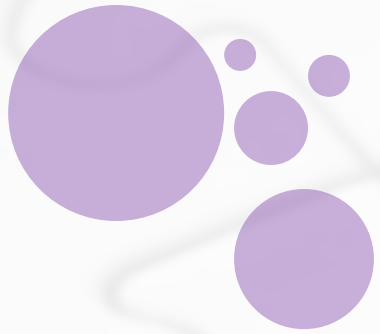
    system( "pause" );
    return 0;
}
```



Linked list

```
printf("%d %d %d \n",p1->num,p1->next->num,p1->next->next->num);
```



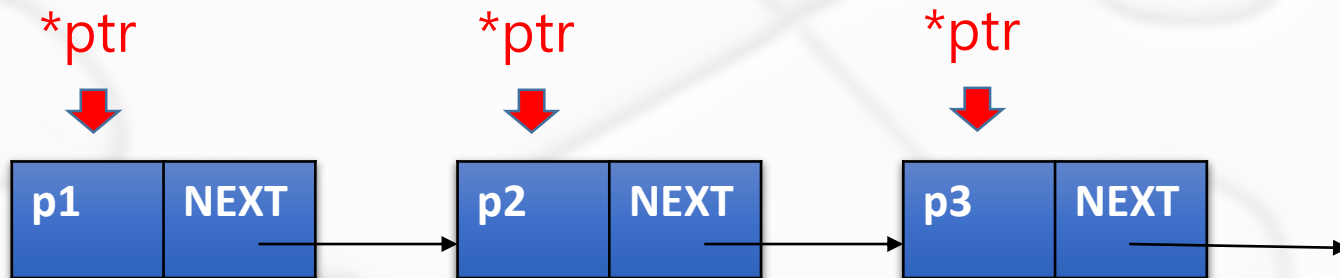


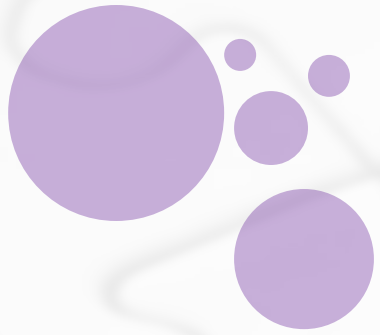
Linked list – 移動

```
node *ptr = p1;  
printf("%d",ptr->num);
```

```
ptr = ptr->next;  
printf("%d",ptr->num);
```

```
ptr = ptr->next;  
printf("%d",ptr->num);
```



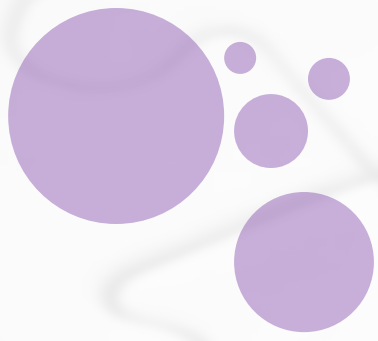


Linked list – 移動

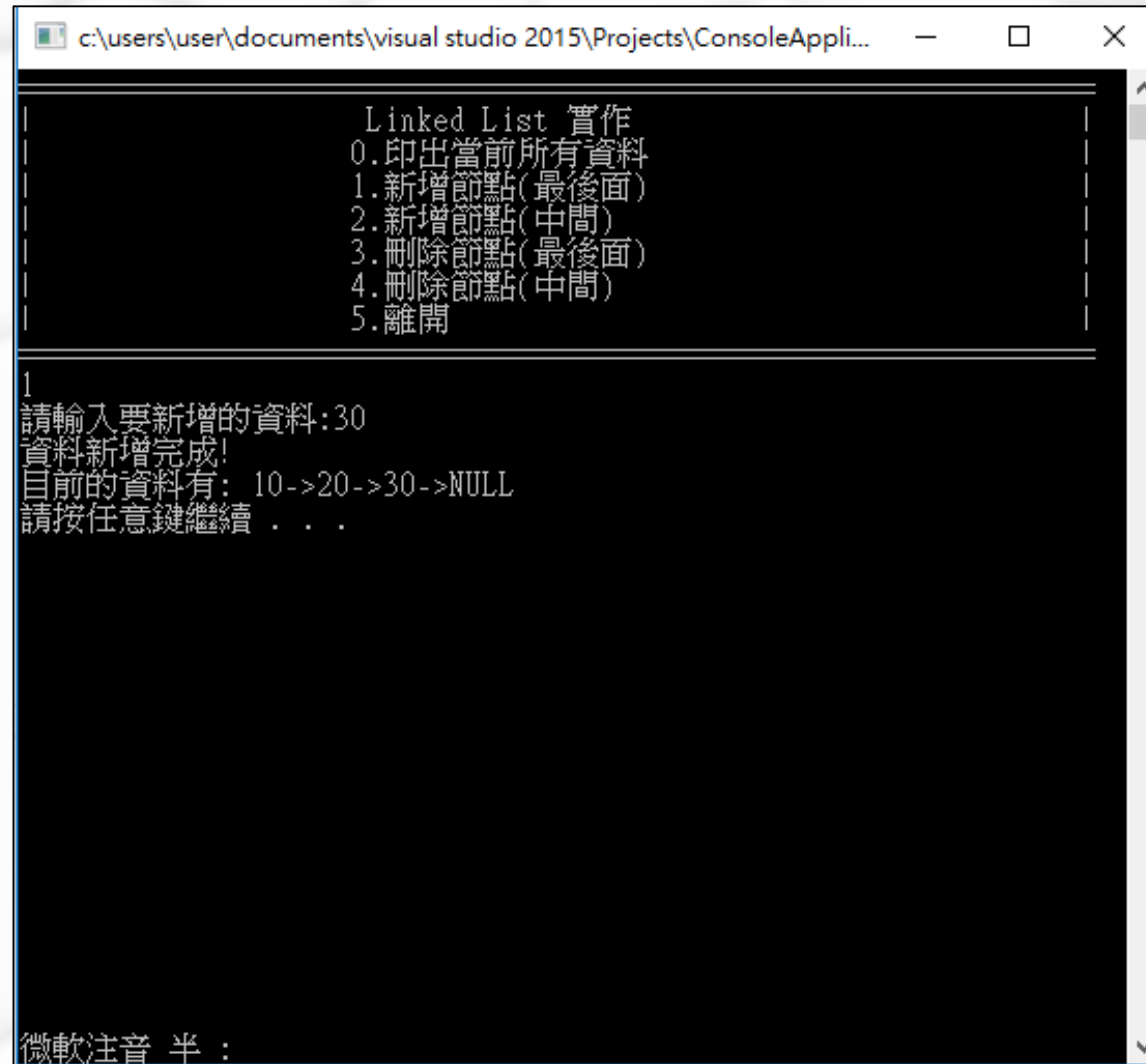
寫成迴圈

```
node *ptr = p1;  
while(ptr != NULL)  
{  
    printf("%d \n",ptr->num);  
    ptr = ptr->next;  
}
```





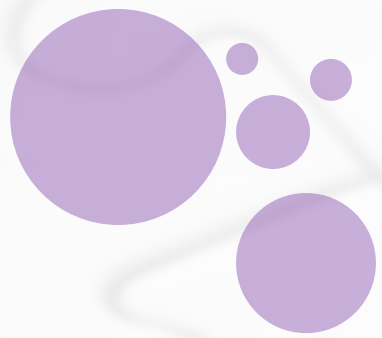
用Linked list做資料新增、刪除



```
c:\users\user\documents\visual studio 2015\Projects\ConsoleAppli...
Linked List 實作
0.印出當前所有資料
1.新增節點(最後面)
2.新增節點(中間)
3.刪除節點(最後面)
4.刪除節點(中間)
5.離開

1
請輸入要新增的資料:30
資料新增完成!
目前的資料有: 10->20->30->NULL
請按任意鍵繼續 . . .

微軟注音 半 :
```



Linked List Structure & Head

```
struct node {  
    int num;  
    node* next;  
};
```

```
int main()  
{  
    //initial head node  
    node *head = new node;  
    head->num = 0;  
    head->next = NULL;  
    int input = 0;  
    int new_data = 0;  
  
    while (1) { ... }  
  
    return 0;  
}
```

有沒有玩過老鷹抓小雞？

```
int main()
{
    //initial head node
    node *head = new node;
    head->num = 0;
    head->next = NULL;
    int input = 0;
    int new_data = 0;

    while (1) { ... }

    return 0;
}
```



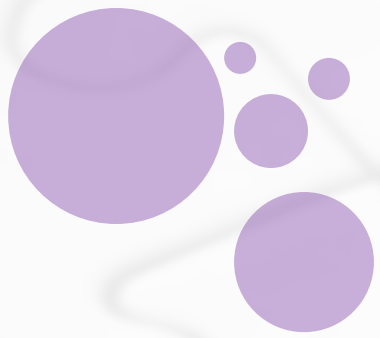
Head節點紀錄串列的開頭，
習慣上不會拿來存資料，而
是當作一個開頭節點。



```

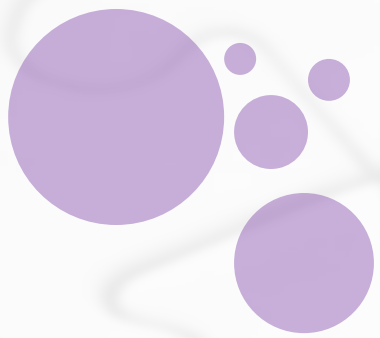
while (1)
{
    printf("=====\\n");
    printf("|           Linked List 實作           |\\n");
    printf("|           0.印出當前所有資料           |\\n");
    printf("|           1.新增節點(最後面)           |\\n");
    printf("|           2.新增節點(中間)             |\\n");
    printf("|           3.刪除節點(最後面)           |\\n");
    printf("|           4.刪除節點(中間)             |\\n");
    printf("|           5.離開                       |\\n");
    printf("=====\\n");
    scanf("%d", &input);
    switch (input)
    {
        case 0:
            print(head);
            break;
        case 1:
            printf("請輸入要新增的資料:");
            scanf("%d", &new_data);
            insert_to_end(head, new_data);
            break;
        case 2:break;
        case 3:
            delete_end(head);
            break;
        case 4:break;
        case 5:
            exit(0);
            break;
    }
    system("pause");
    system("cls");
}

```



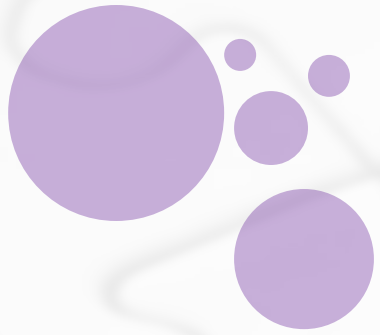
print()

```
void print(node* head)
{
    node* ptr = head;
    if (ptr->next == NULL)printf("目前Linked list中沒有資料!\n");
    else
    {
        printf("目前的資料有: ");
        while (ptr->next != NULL)
        {
            ptr = ptr->next;
            printf("%d->", ptr->num);
        }
        printf("NULL\n");
    }
}
```

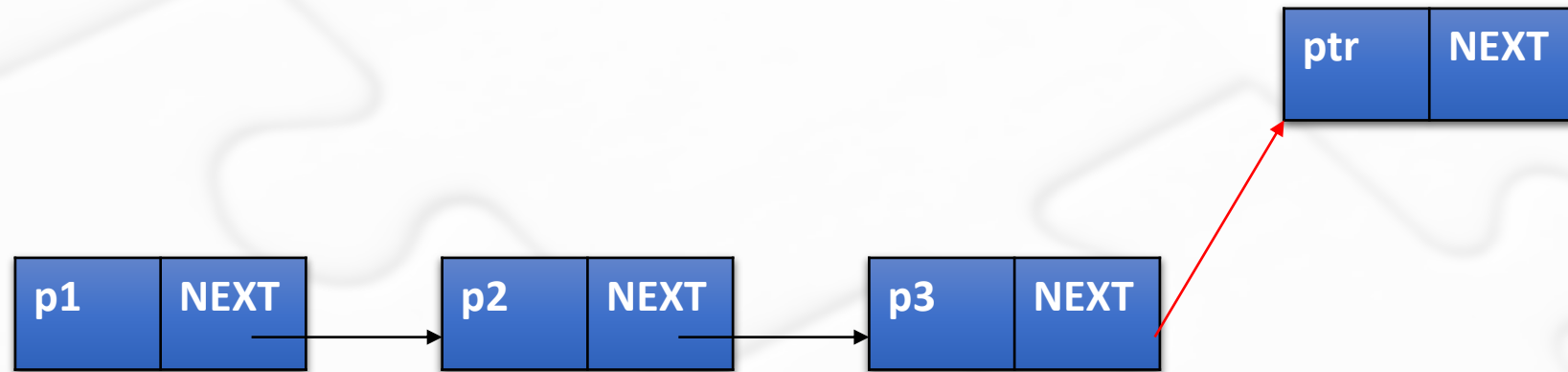


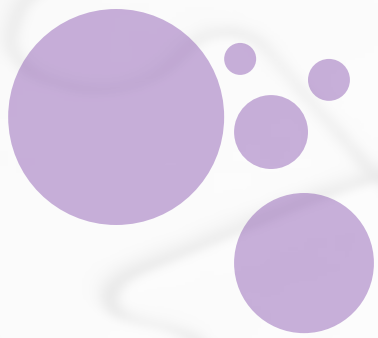
insert_to_end()

```
void insert_to_end(node* head, int data)
{
    node* ptr = head;
    while (ptr->next != NULL)
    {
        ptr = ptr->next;
    }
    //新增節點在尾端
    node* newptr = new node;
    newptr->num = data;
    newptr->next = NULL;
    ptr->next = newptr;
    printf("資料新增完成!\n");
    print(head);
}
```

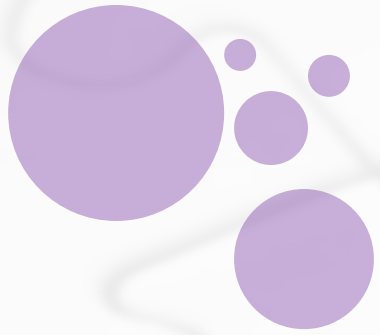
Linked list – 新增(2)





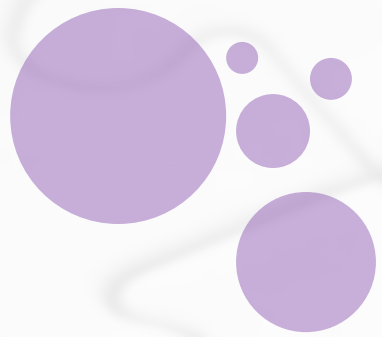
delete_end()

```
void delete_end(node* head)
{
    node* ptr = head;
    if (ptr->next == NULL)printf("沒有資料可以刪除!\n");
    else
    {
        while (ptr->next->next != NULL) //移到NULL前兩個node
        {
            ptr = ptr->next;
        }
        node* delete_node = ptr->next;
        ptr->next = NULL;
        delete delete_node;
        printf("資料刪除完成\n");
        print(head);
    }
}
```



Linked list – 刪除(2)

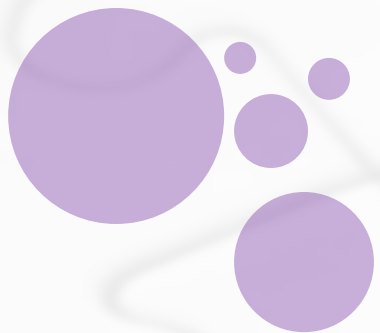




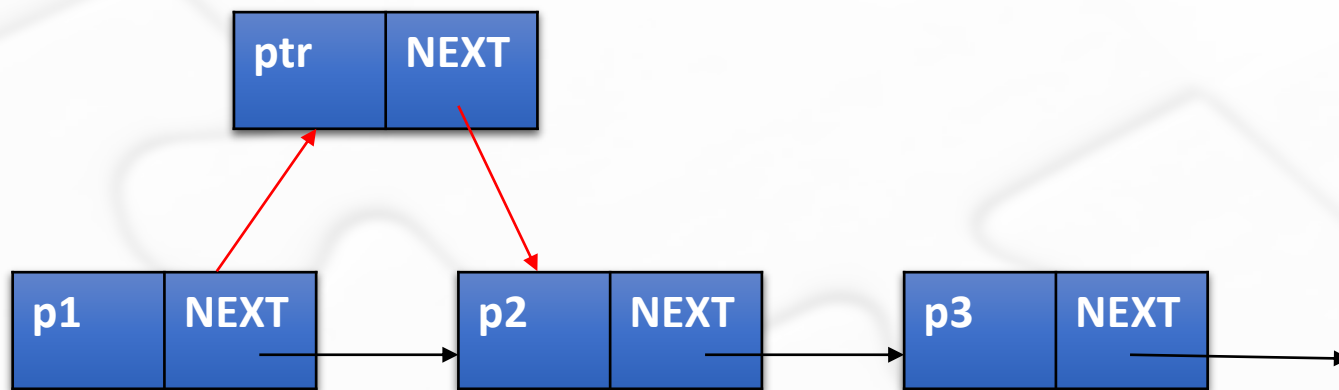
Others function..

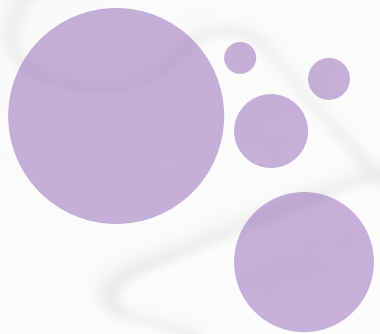
```
void insert_at_position(node* head, int position, int data)
{
    //DO IT YOURSELF
}
```

```
void delete_at_position(node* head, int position)
{
    //DO IT YOURSELF
}
```



Linked list – 新增





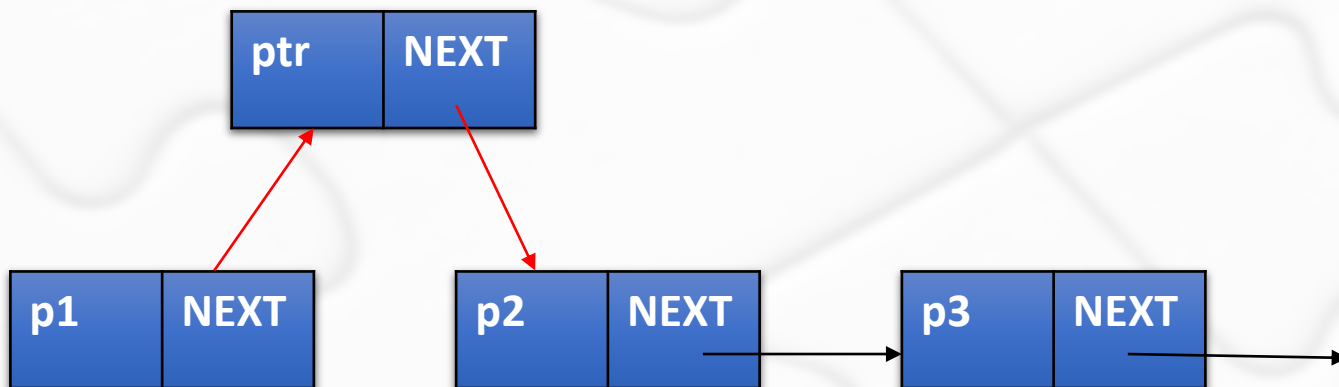
Linked list – 新增

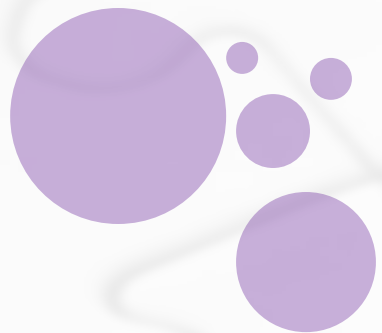
```
node *ptr = new node;//新增一個節點
```

```
ptr->num = 40;
```

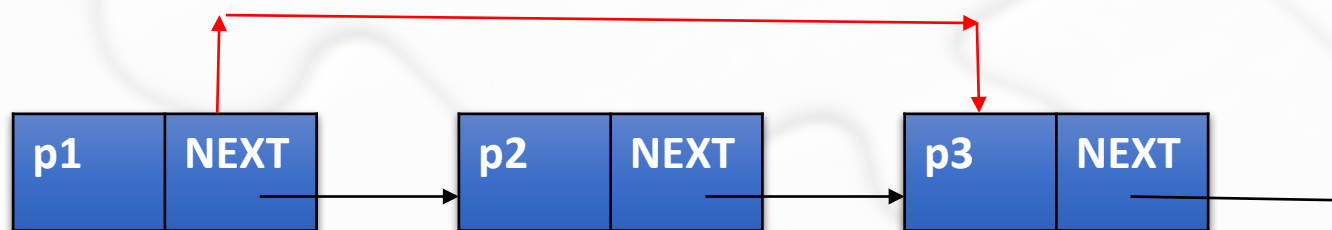
```
p1->next = ptr;
```

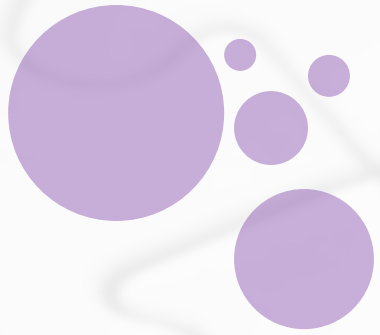
```
ptr->next = p2;//這是錯的!!我要怎麼知道p2在哪裡?
```





Linked list – 删除

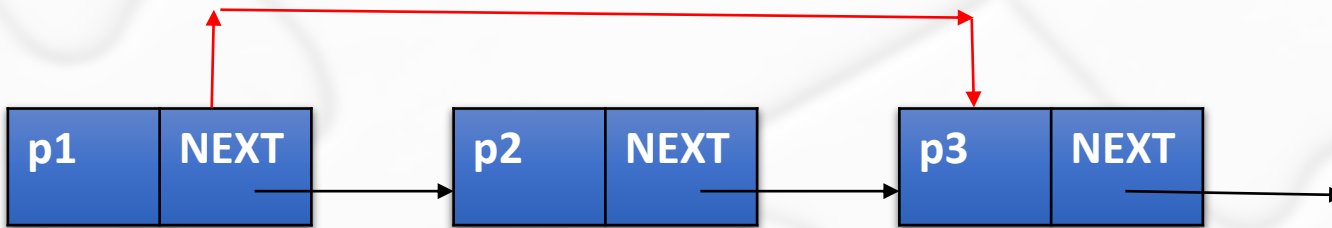


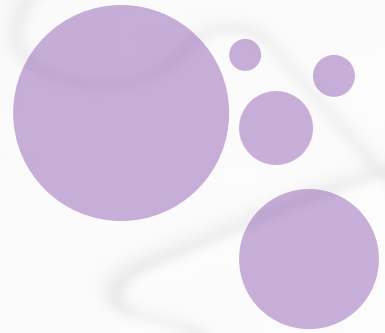


Linked list – 刪除

```
node *ptr = p1->next;  
p1->next = p1->next->next;
```

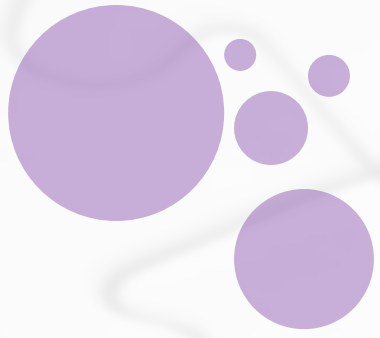
delete ptr;//刪除ptr所在位置的節點

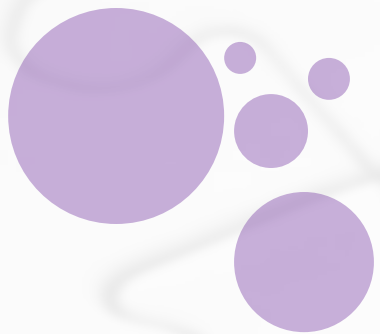




Linked List vs. Array?

| | Linked List | Array |
|-----------|-------------|--------|
| 讀取方式 | 循序讀取 | 隨機存取 |
| 尋找特定位置的資料 | $O(N)$ | $O(1)$ |
| 新增、刪除 | 很簡單 | 很麻煩 |
| 動態大小 | 我有 | 我沒有QQ |



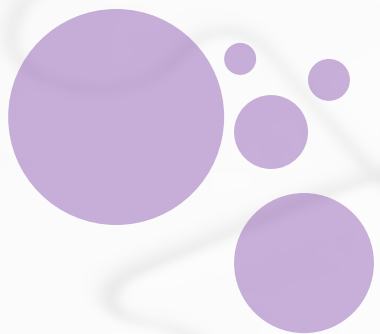


資料結構

Stack(堆疊)

Queue(佇列)





Stack

堆疊(Stack)：

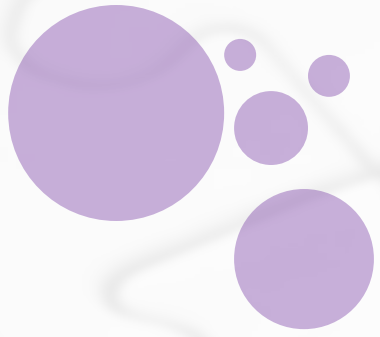
- ▣ 資料結構的一種

- ▣ LIFO (Last In, First Out)

- ▣ Ex：疊盤子的時候，一定是從最上層(最後放的)開始拿

- ▣ 程式的呼叫順序就是一種Stack

- ▣ 遞迴的原理也是Stack



Stack

基本操作：

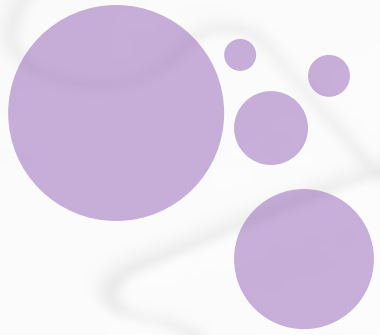
 push

 pop

 top

 size

 empty

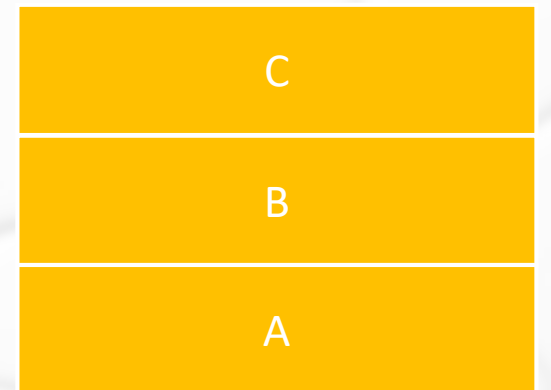


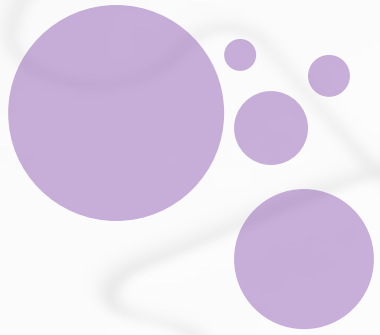
Stack

基本操作：

 push

 push A -> push B -> push C



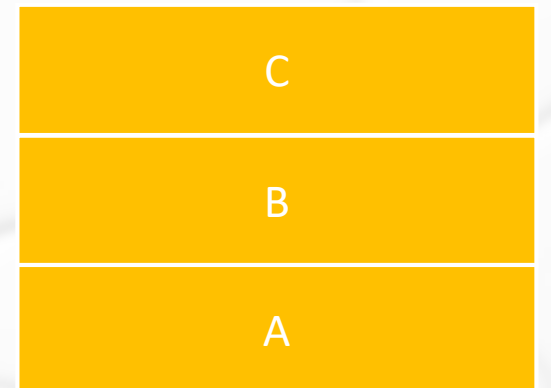


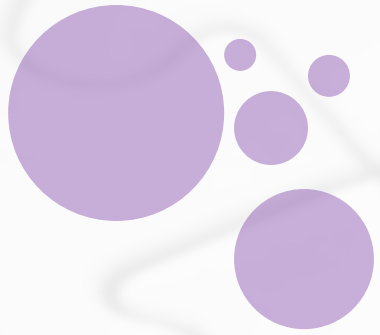
Stack

基本操作：

 pop

 pop -> pop -> pop

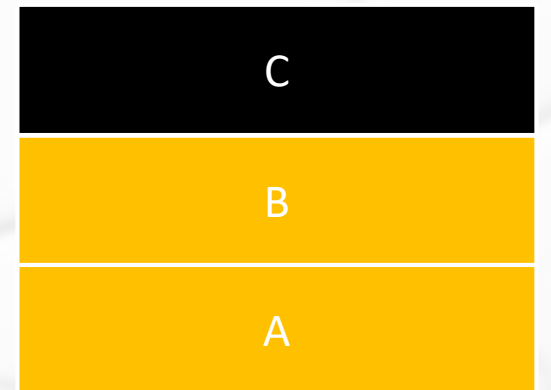


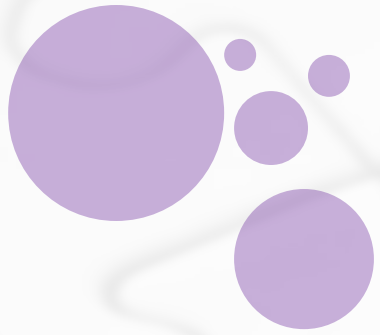


Stack

基本操作：

 top

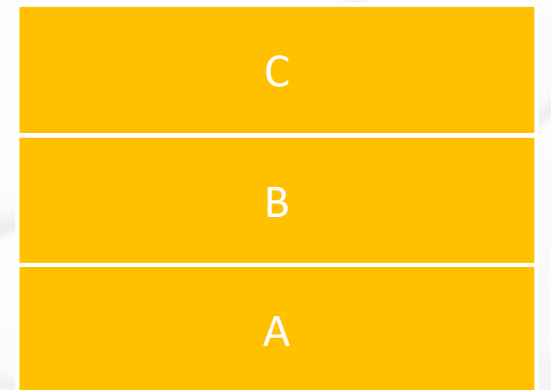


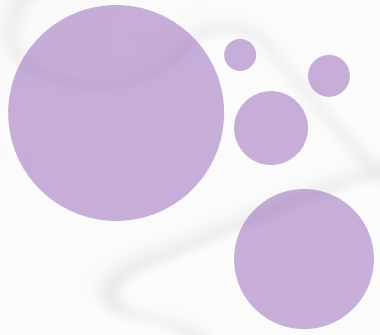


Stack

基本操作：

 size = 3

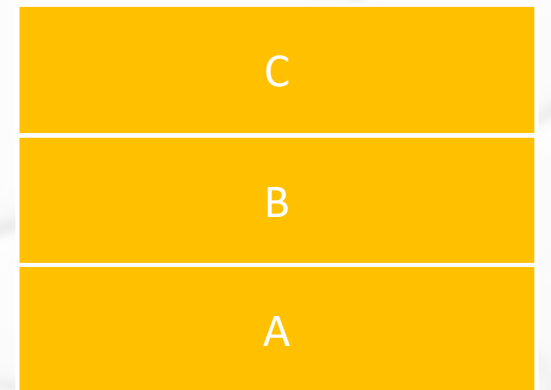


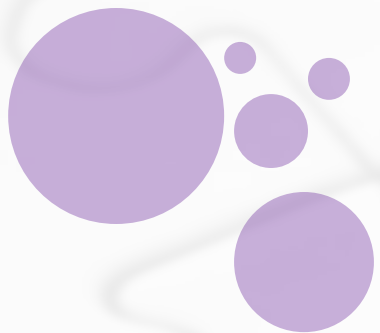


Stack

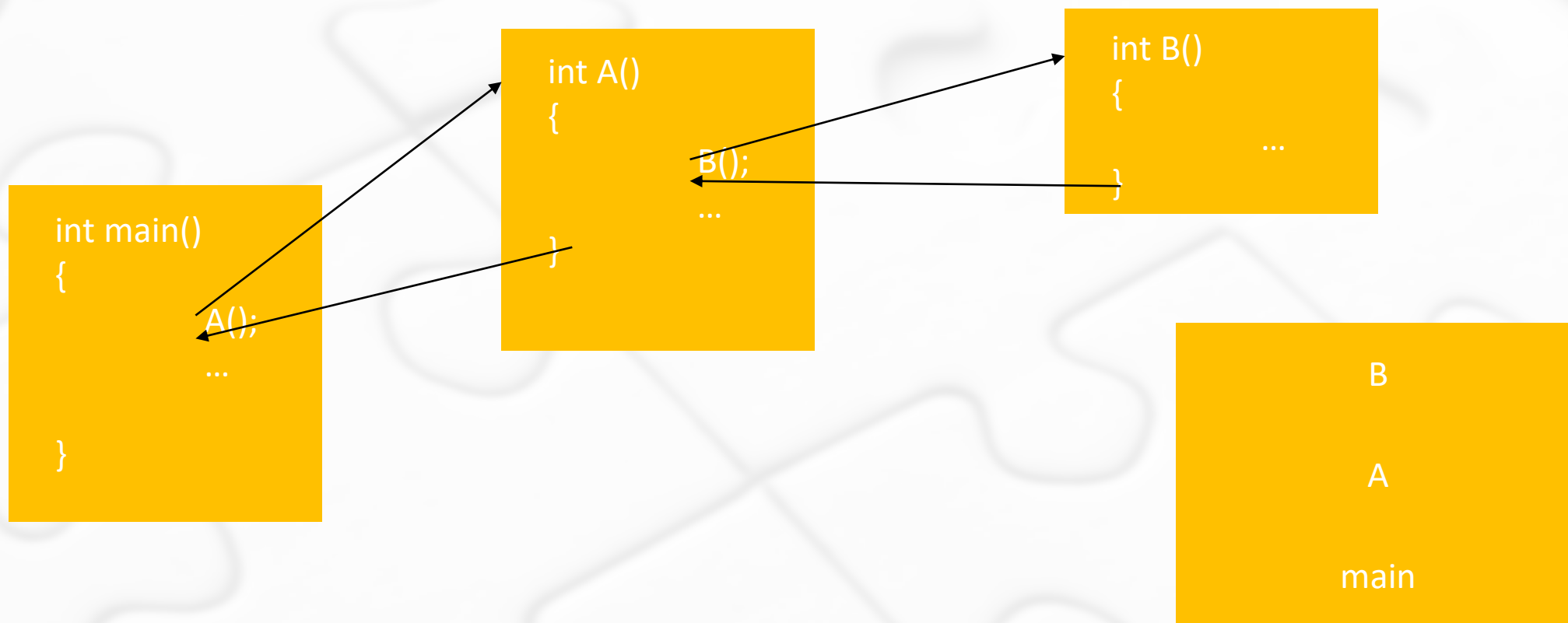
基本操作：

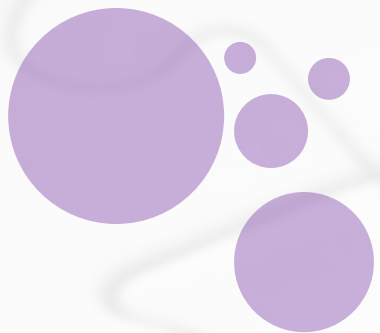
 empty



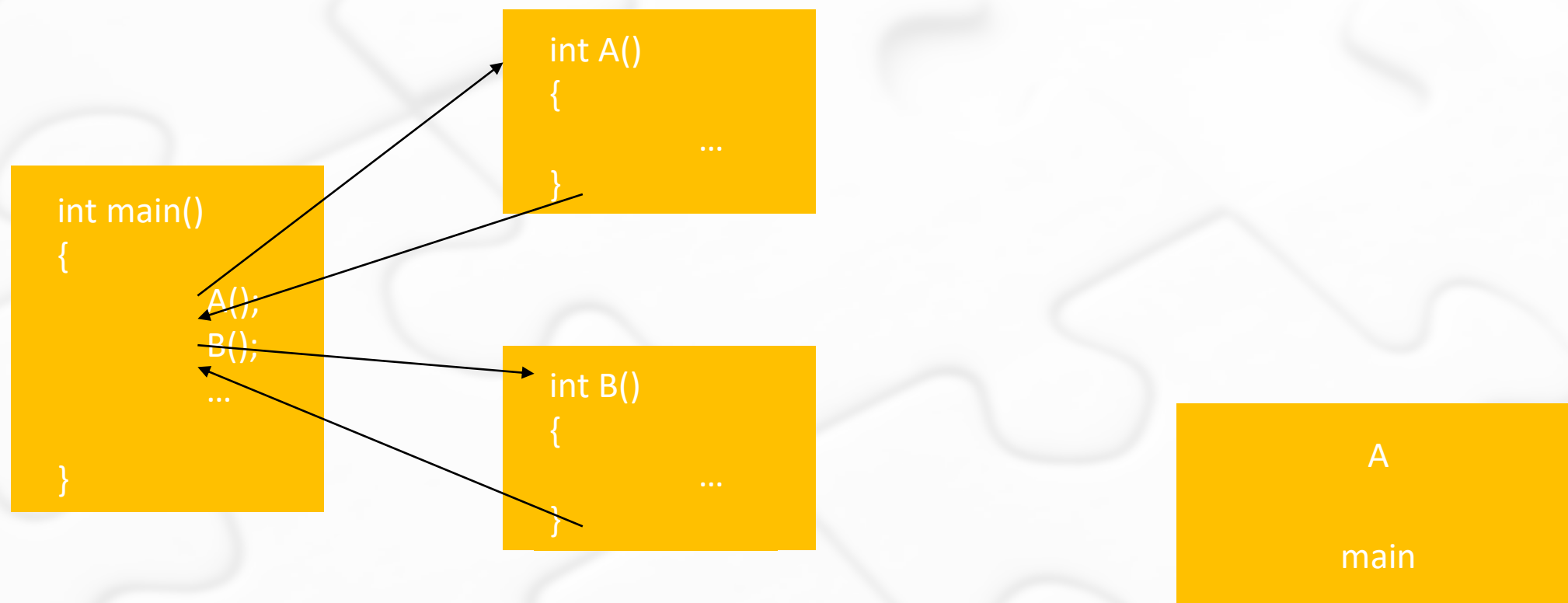


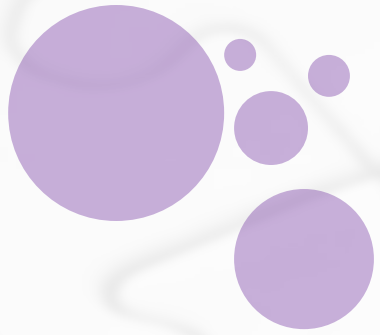
程式的執行順序





程式的執行順序





Queue

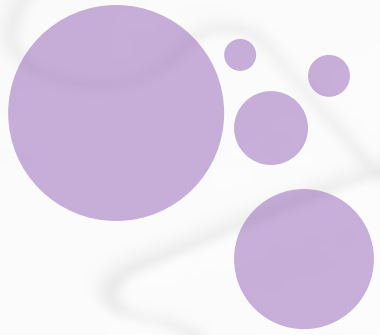
佇列(Queue)：

- ▣ 資料結構的一種

- ▣ **FIFO(First In,First Out)**

- ▣ Ex：排隊的時候，最先排的最先離開

- ▣ CPU排程、廣度優先演算法(BFS)



Queue

基本操作：

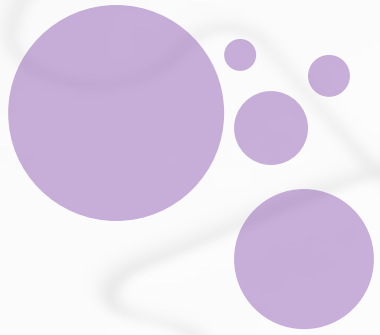
 push

 pop

 front

 size

 empty

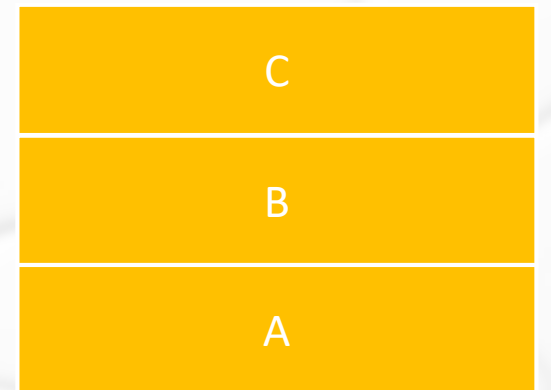


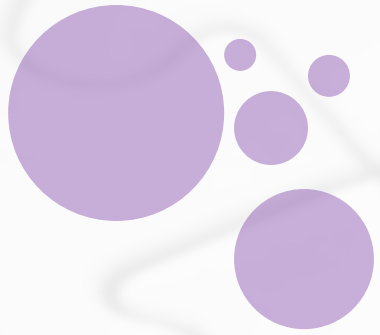
Queue

基本操作：

 push

 push A -> push B -> push C





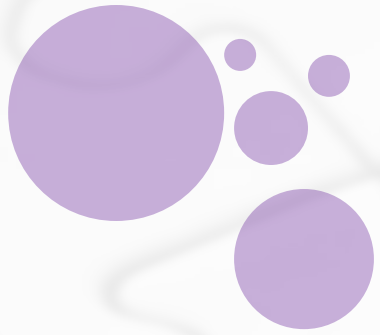
Queue

基本操作：

 pop

 pop -> pop -> pop

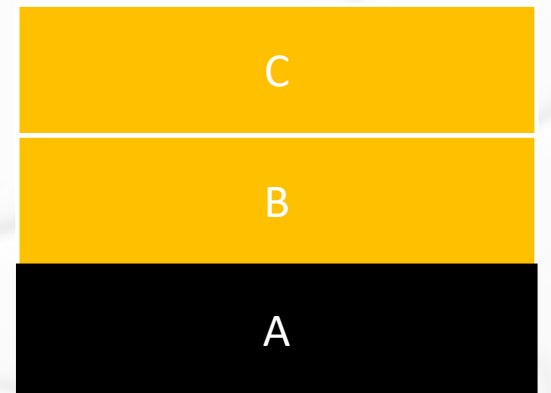
| |
|---|
| C |
| B |
| A |

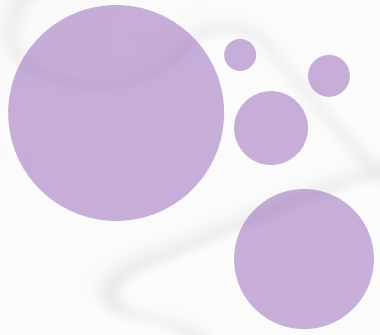


Queue

基本操作：

 front



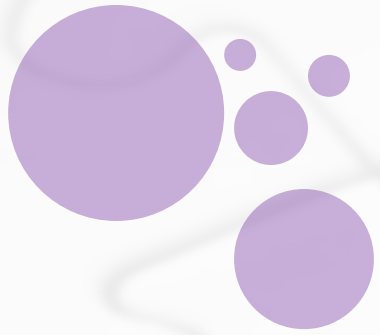


Queue

基本操作：

 size = 3

| |
|---|
| C |
| B |
| A |

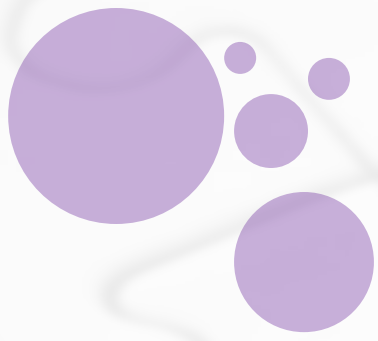


Queue

基本操作：

 empty

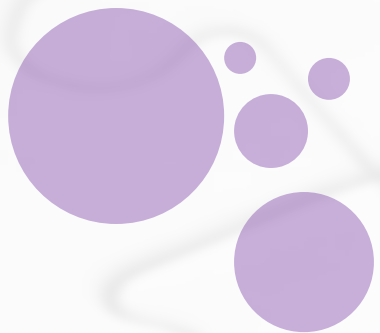
| |
|---|
| C |
| B |
| A |



老鼠走迷宮

❖ 有一個迷宮，在終點（右下角）放置老鼠最愛吃的奶酪，將老鼠放在左上角，老鼠必須穿越迷宮才能找到他的食物，請問老鼠該怎麼走？

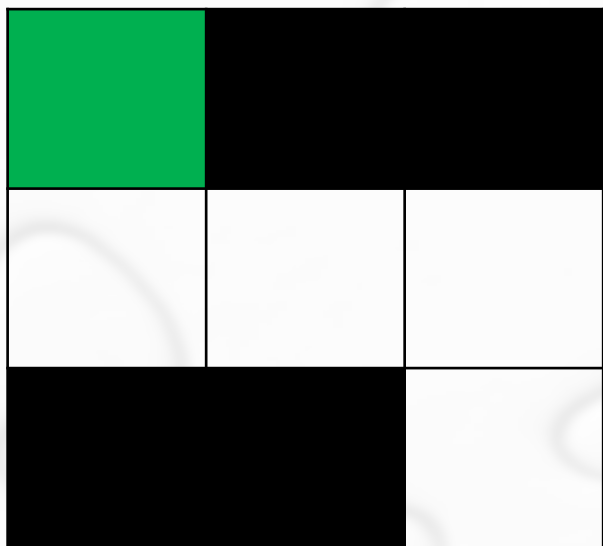




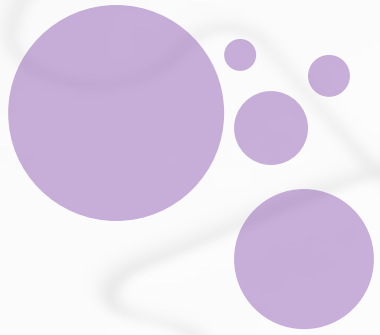
老鼠走迷宮

🐭 我們怎麼走迷宮的？

🐭 往一個方向走，直到沒路了折返後再走另外一個方向

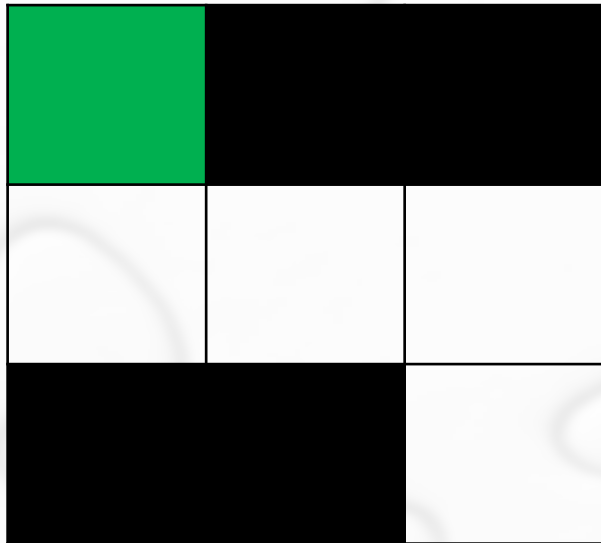


Q：假設走的先後順序依序是：
右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？

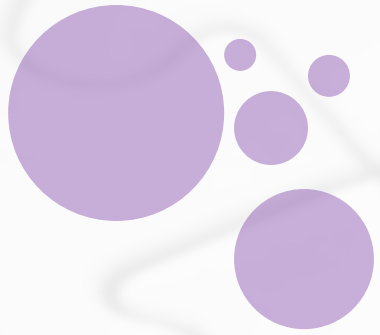


老鼠走迷宮

Q：假設走的先後順序依序是：右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？



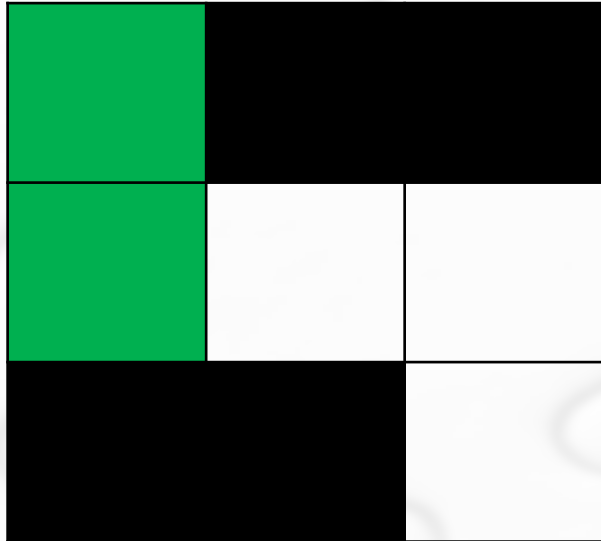
- 1.向右(X)
- 2.向左(X)
- 3.向下(O)

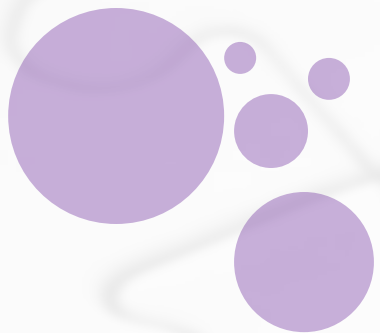


老鼠走迷宮

Q：假設走的先後順序依序是：右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？

1.向右(O)

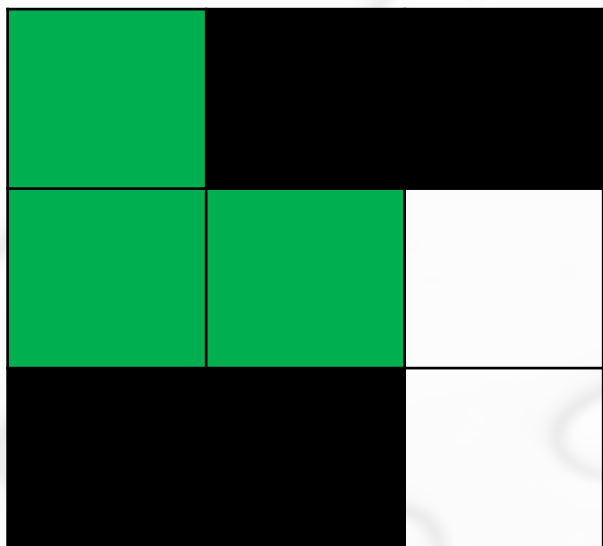


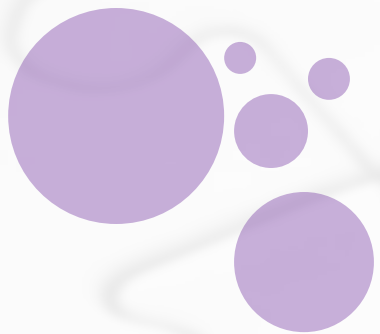


老鼠走迷宮

Q：假設走的先後順序依序是：右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？

1.向右(O)

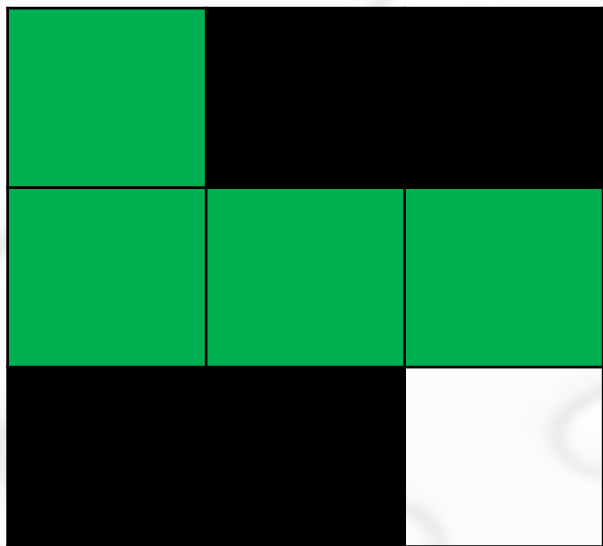


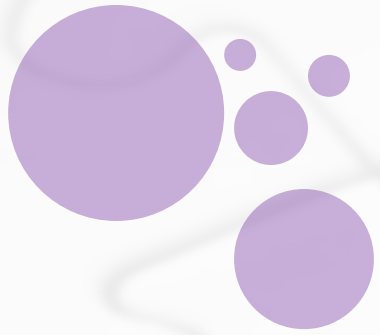


老鼠走迷宮

Q：假設走的先後順序依序是：右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？

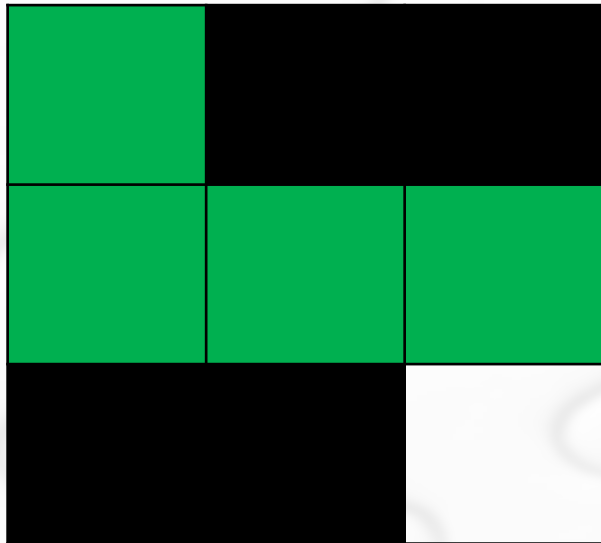
1.向右(O)



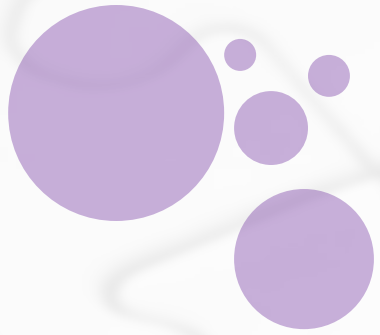


老鼠走迷宮

Q：假設走的先後順序依序是：右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？

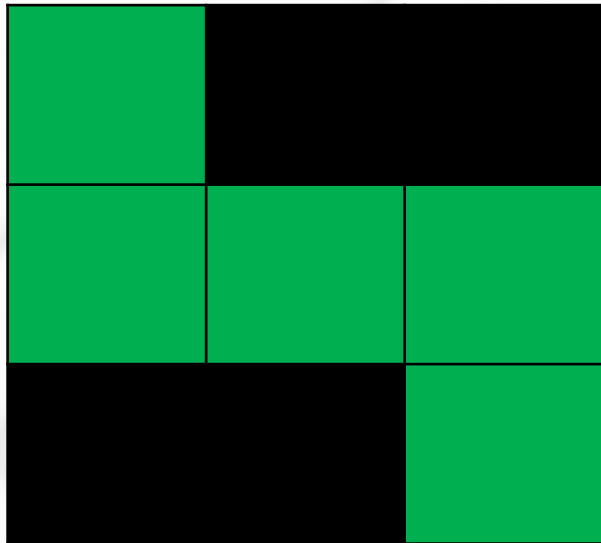


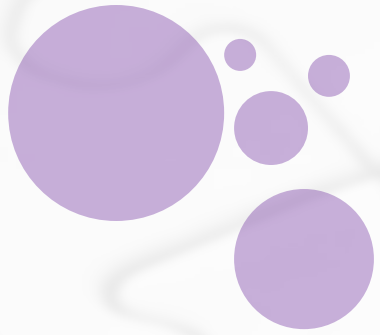
- 1.向右(X)
- 2.向左(X) -> 走過的就不能再走了，why?
- 3.向下(O)



老鼠走迷宮

Q：假設走的先後順序依序是：右邊->左邊->下面->上面
這張3*3的地圖會怎麼走？

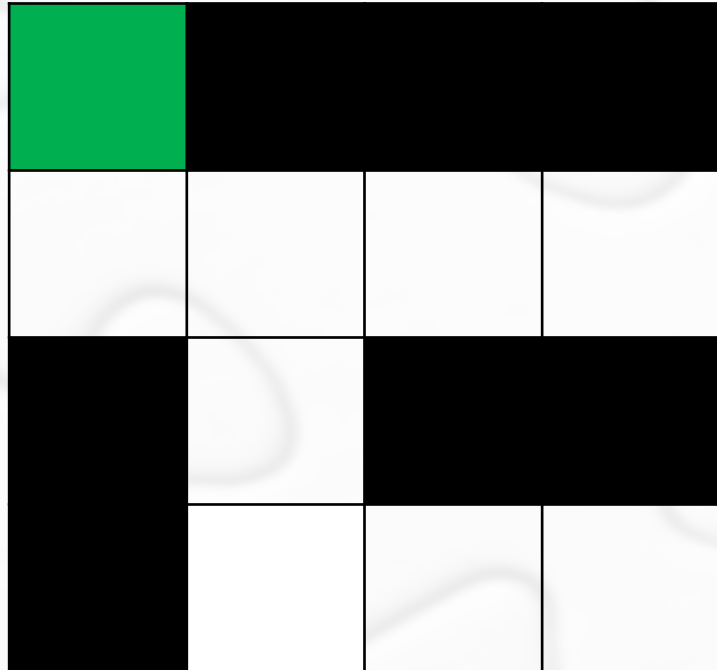




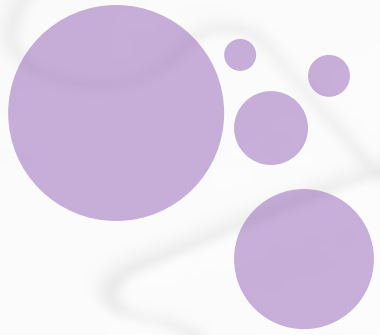
老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



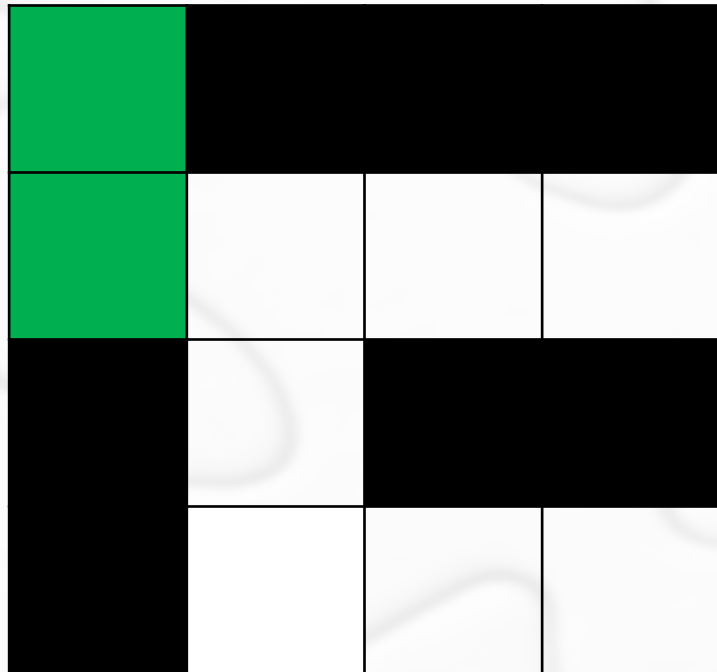
- 1.向右(X)
- 2.向左(X)
- 3.向下(O)



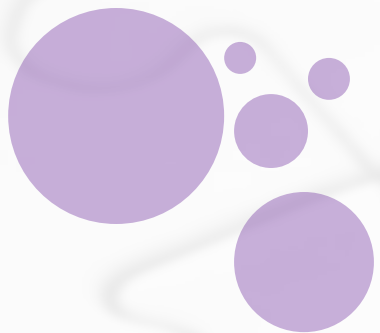
老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



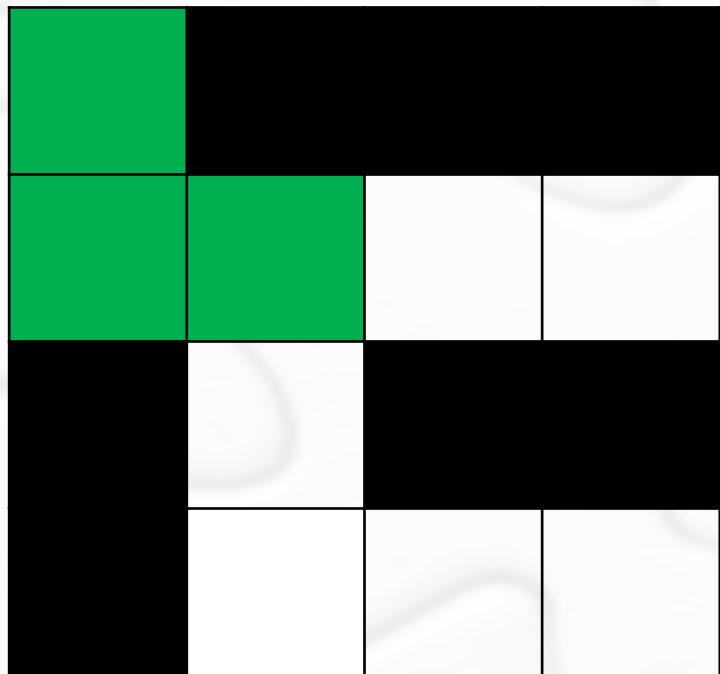
1.向右(O)



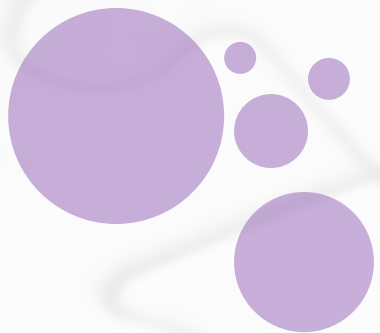
老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



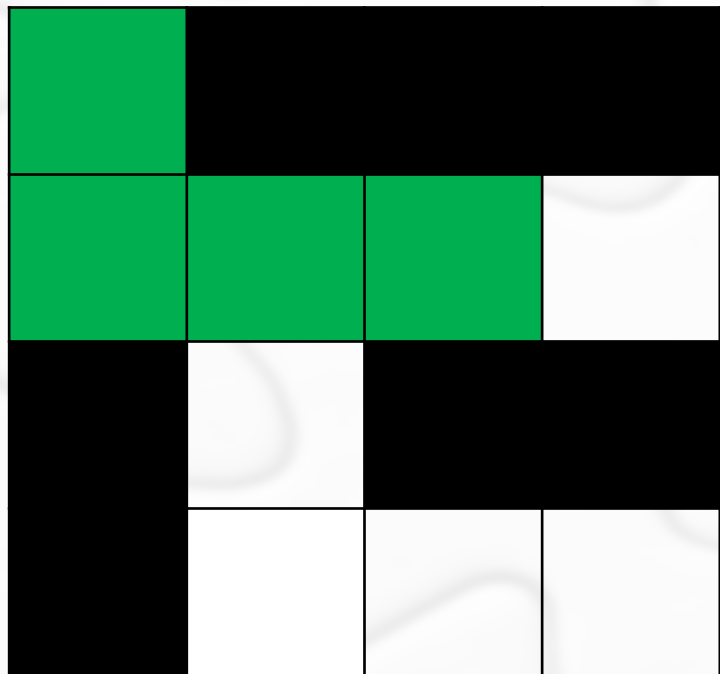
1.向右(O)



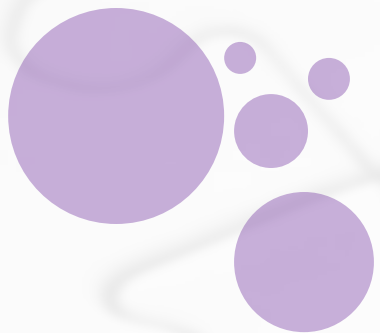
老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



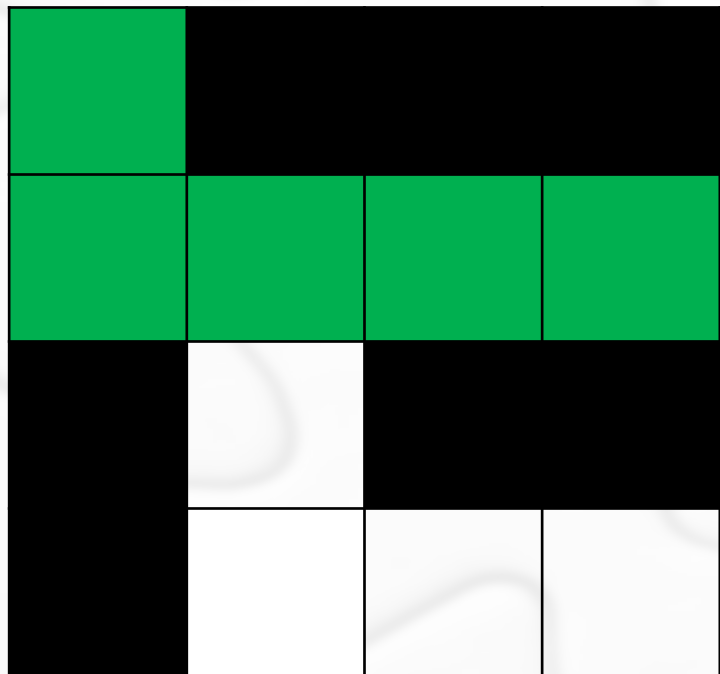
1.向右(O)



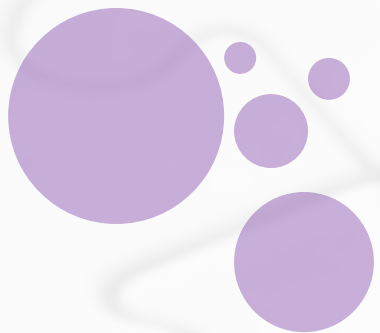
老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



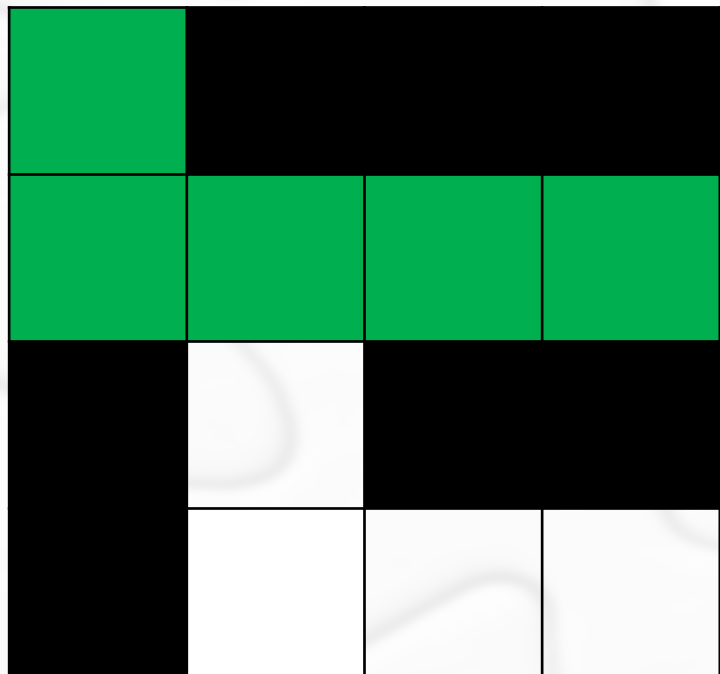
1.向右(O)



老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



1.向右(X)

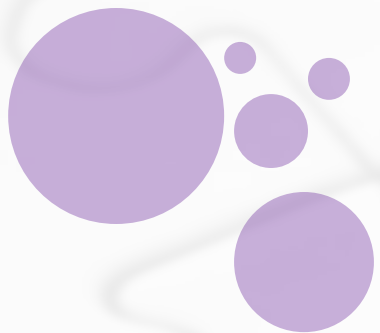
2.向左(X)

3.向下(X)

4.向上(X)

都不能走了怎麼辦？

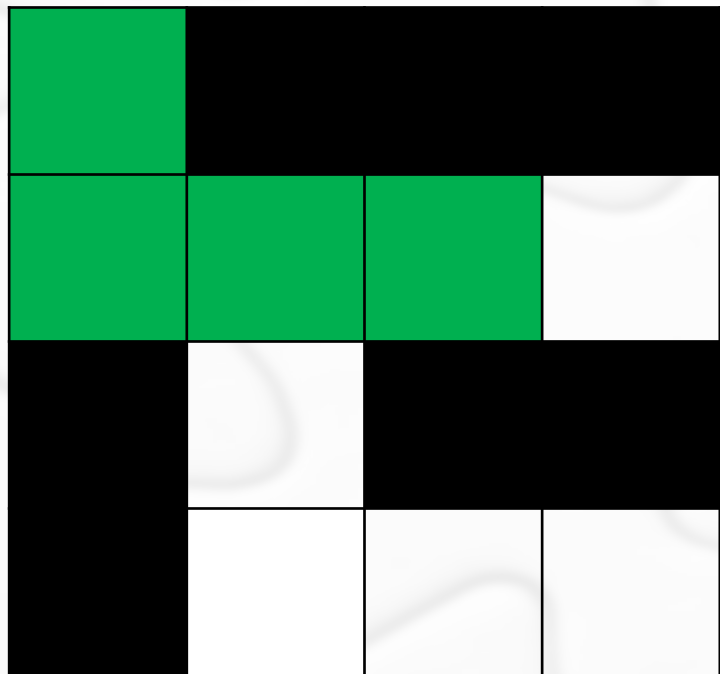
=>代表這一步是錯的，退回前一步



老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



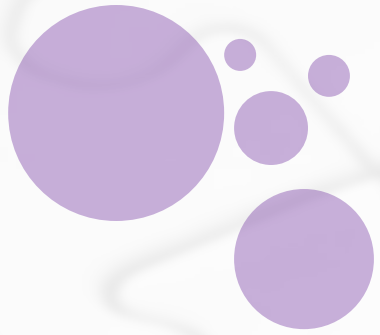
1.向右(O)

2.向左(X)

3.向下(X)

4.向上(X)

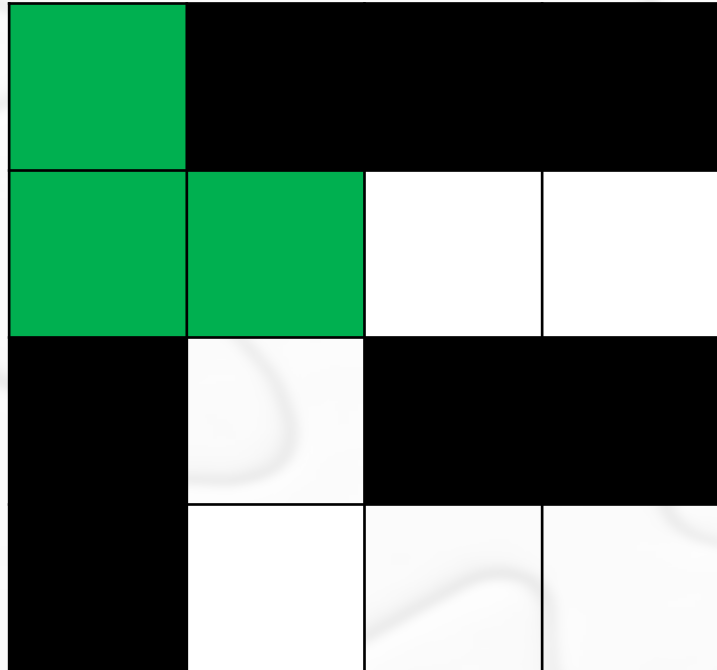
=> 退回前一步



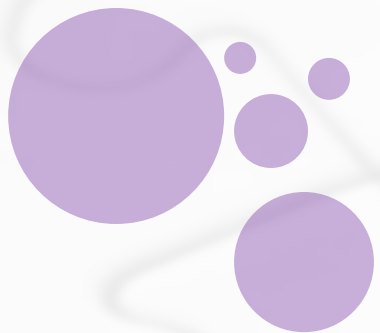
老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？



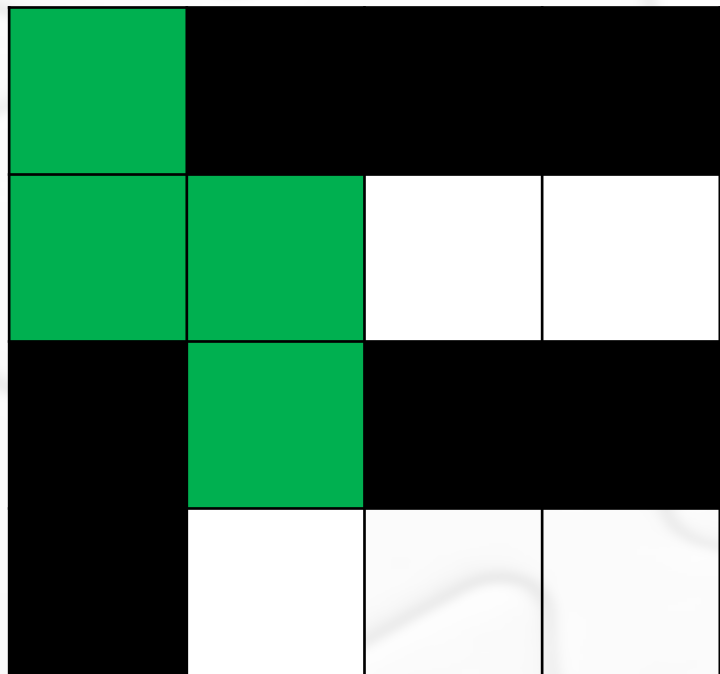
- 1.向右(O)
- 2.向左(X)
- 3.向下(O)

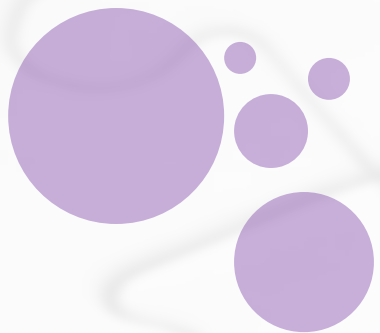


老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？

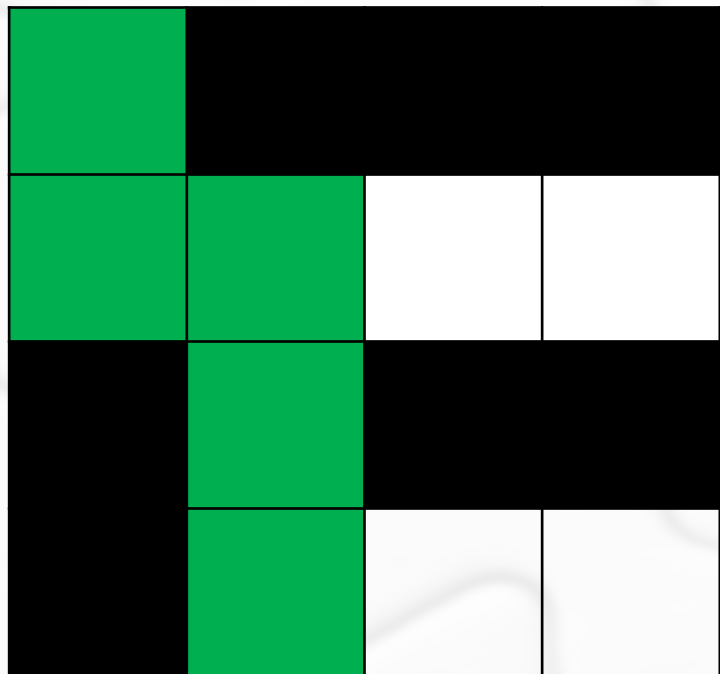


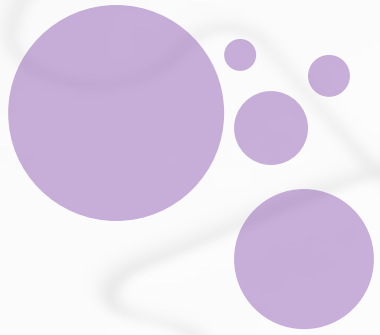


老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？

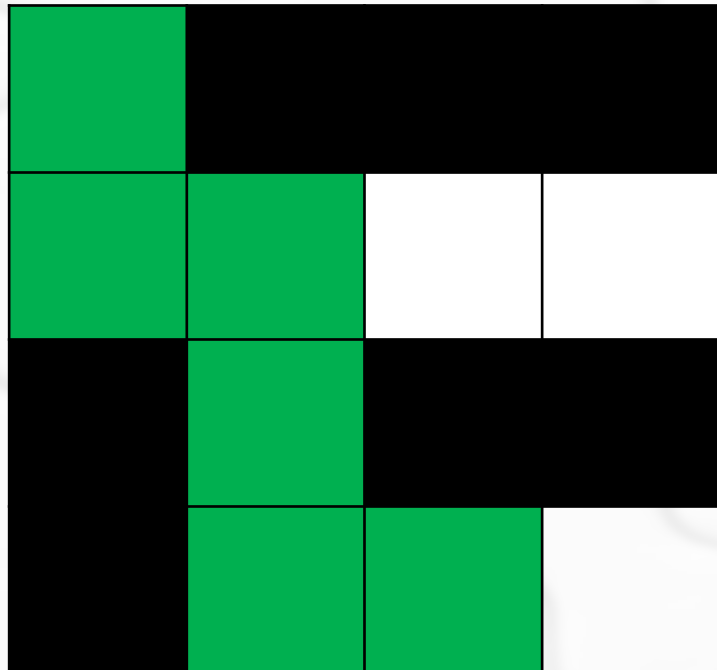


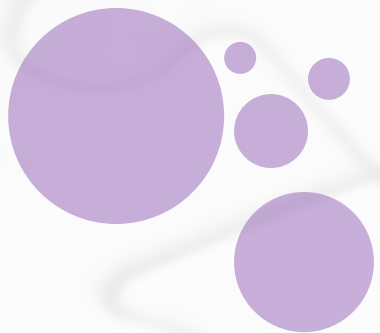


老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？

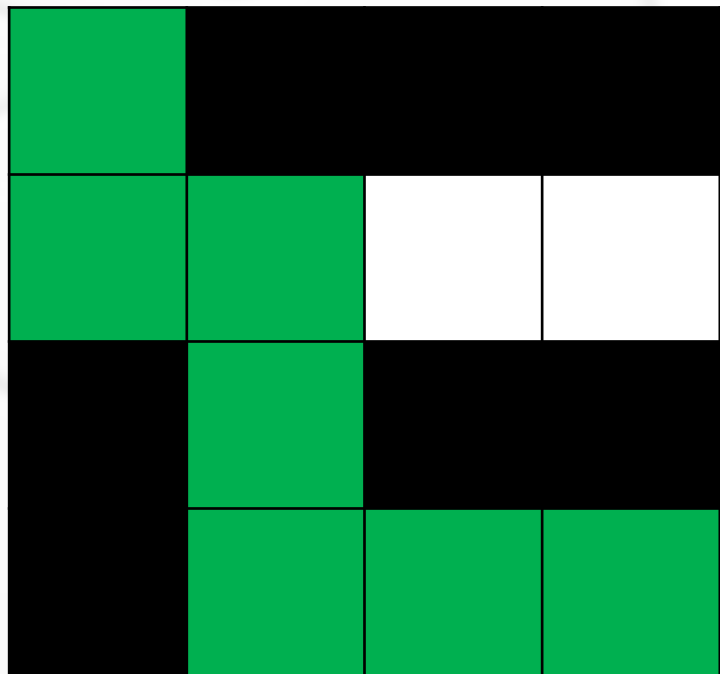


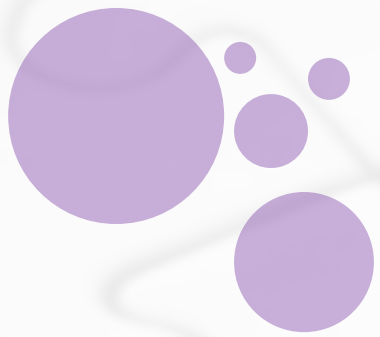


老鼠走迷宮

Q2：假設走的先後順序依序是：右邊->左邊->下面->上面

這張4*4的地圖會怎麼走？





HW4-1

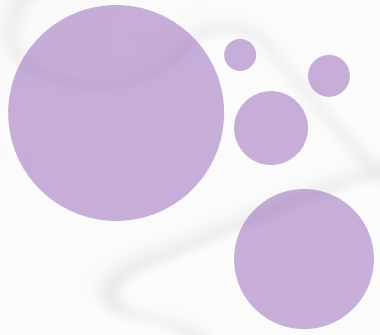
- 使用Linked List實作Stack & Queue
- 介面呈現類似Linked list練習，需可讓使用者輸入數字執行對應的指令

->

```
Linked List 實作
0.印出當前所有資料
1.新增節點(最後面)
2.新增節點(中間)
3.刪除節點(最後面)
4.刪除節點(中間)
5.離開

請輸入要新增的資料:30
資料新增完成!
目前的資料有: 10->20->30->NULL
請按任意鍵繼續...
```

- 必須包含上課講到的5項操作
- Structure名稱必須是Stack和Queue
- Stack和Queue各為一個專案



HW4-2

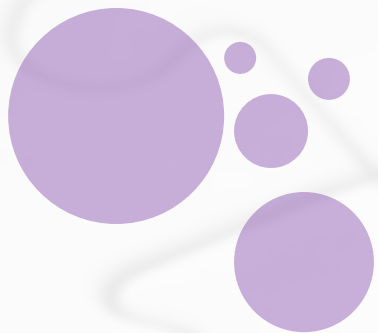
老鼠走迷宮，規則：

❖ 走路順序: 右->上->左->下

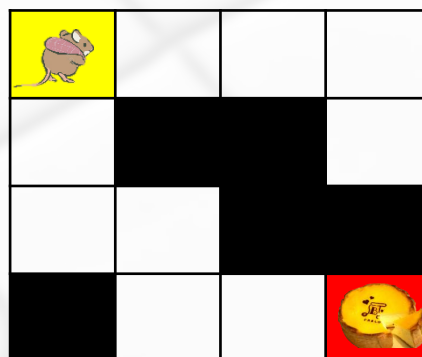
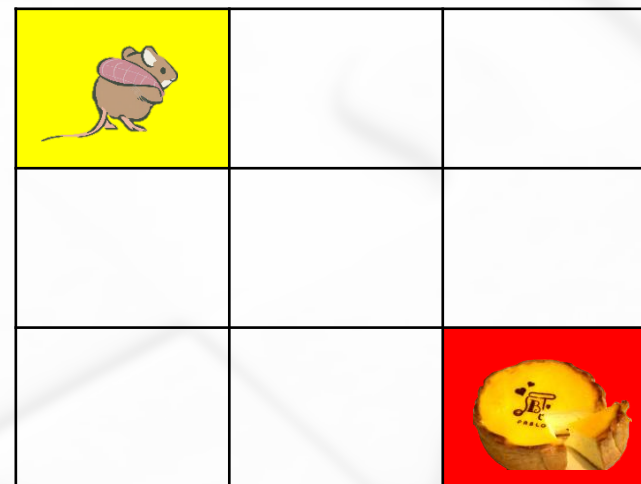
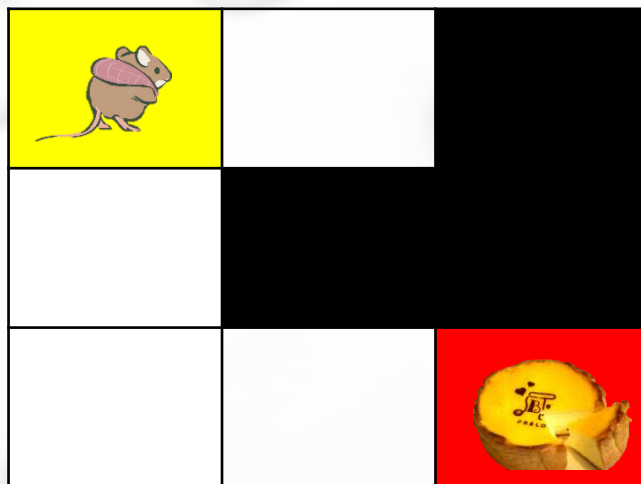
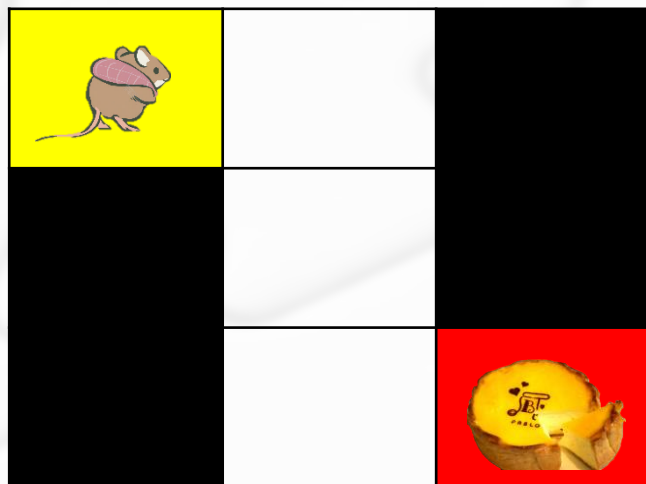
❖ 走過的路不再走，如果沒有路了則回到上一步

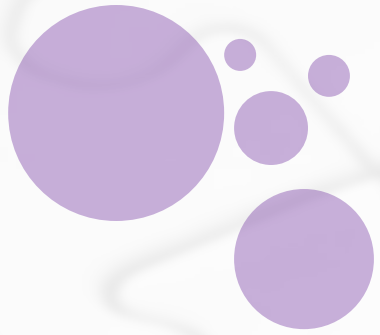
❖ 直到走到終點

請用紙和筆將每一步驟的順序寫出來

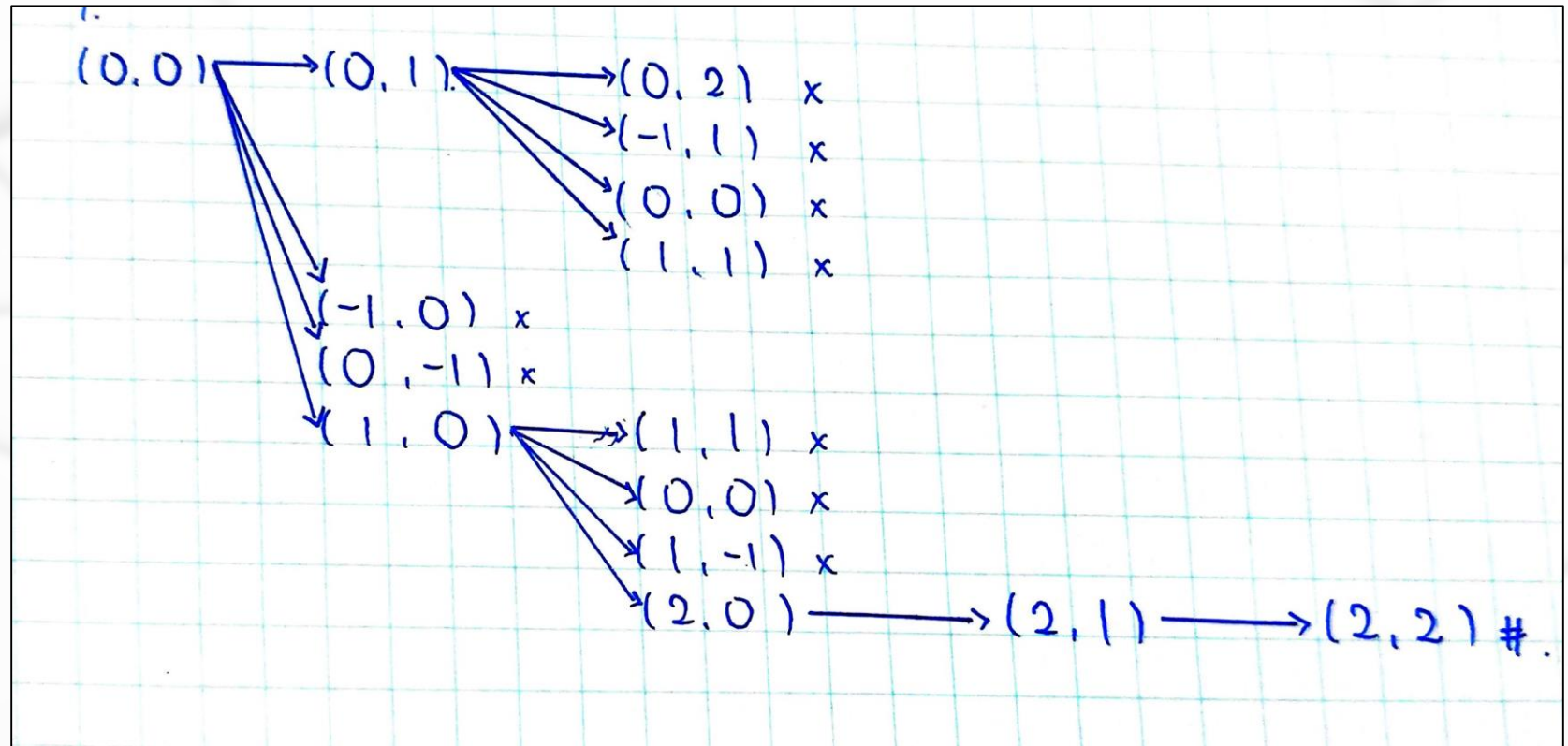
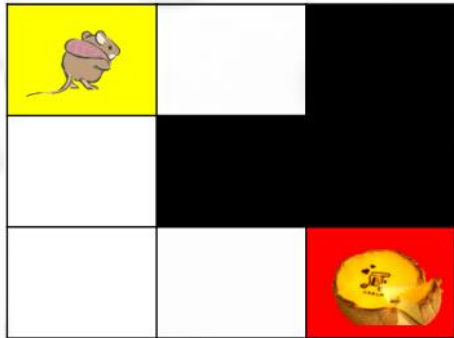


HW4-2





HW4-2 範例





THANK YOU