

MODULE 3 –LIST, STACK AND QUEUE



1. โครงสร้างข้อมูลแบบ **list** สายการ
2. การใช้ Array สร้าง **list** และ **Operation**
3. การใช้ pointer สร้าง **Linked list** และ **operation**
4. การใช้ pointer สร้าง **Doubly linked list** และ **Operation**
5. การใช้ pointer สร้าง **Circular linked list**
6. **Stack** และ **Operation**
7. **Queue** และ **Operation**



เนื้อหาที่เรียนในวันนี้

เมื่อเรียนจบแล้ว สิ่งที่คุณจะสามารถทำได้คือ

- 1)สามารถแยกแยะระหว่าง list และ linked list ได้
- 2)เข้าใจการทำงานของ linked list
- 3)สามารถเขียนโปรแกรมทำการเก็บข้อมูลลงใน linked list ได้
- 4)สามารถเขียนโปรแกรมทำการลบข้อมูลลงใน linked list ได้
- 5)สามารถเขียนโปรแกรมทำการค้นหาข้อมูลลงใน linked list ได้
- 6)สามารถเขียนโปรแกรมทำการนับจำนวนข้อมูลลงใน linked list ได้
- 7)สามารถนำ linked list ไปใช้ในการปัญหาทางการโปรแกรมได้
- 8)ทราบ BigO ทุก Operation



3.1 The List

A General list of the form $A_1, A_2, A_3, \dots, A_N$

For any list ^{ยกเว้น} **except** the empty list, we say that A_{i+1} follows (or succeeds) A_i ($i < N$) and that A_{i-1} ^{ก่อนหน้า} **precedes** A_i ($i > 1$). The first element of the list is A_1 , and the last element A_i in a list is A_N

3.1.1 List Operation

☐ insert

☐ remove

☐ printList

☐ makeEmpty

☐ find



3.1.2 Array List

เขียนง่ายขึ้น น้อยไปรก

1) Insert

1. สร้าง Array โดยคาดคะเนว่า list จะมีข้อมูลมากที่สุดกี่ค่า

```
int a[10];
```

array of integer


2. กำหนดตัวแปร size คือจำนวนข้อมูล pos(position) คือตำแหน่งที่ต้องการแทรก

size = 6 จำนวนข้อมูล
arrsize=10 กำหนด array

0	1	2	3	4	5	6	7
10	20	30	40	50	60		



3. การ insert จะทำโดยการหาตำแหน่งที่ต้องการแทรกข้อมูลก่อน เช่น ต้องการแทรก 35 ตำแหน่งที่จะแทรกคือ 3 ตำแหน่ง เก็บในตัวแปร pos



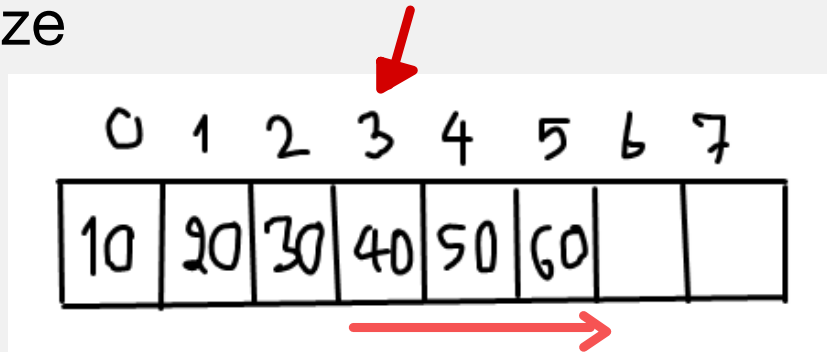
0	1	2	3	4	5	6	7
10	20	30	40	50	60		

```
for(i=0;i<size;i++)
{
    if( data <= a[i])
    {
        pos = i;
        break;
    }
}
```



03603212 : Module3-List, stack, Queue 8

4. จะแทรกได้จะต้องขยับเลื่อนข้อมูลด้านหลังทุกตัวไป 1 ค่า ในช่วง pos-size



เลื่อน : $5 - 3$

size - 1 ตัวแปร pos

เลื่อนข้อมูลใน index 3-5

3

5

pos+1 ถึง size-1

size 6
pos 3

5. ทำการแทรก

```
for (i = size - 1; i >= pos; i--) {  
    a[i+1] = a[i];
```

$a[pos] = 35;$

เอา 5 ใส่ใน 6


```
...
int main()
{ int a[10]={10,20,30,40,50,60};
  int size=6, arrsize=10, i, pos, data;
  cout << "Insert data : ";
  cin >> data;
  for(i=0; i<size; i++)
  { if(a[i]>data)
    {
      pos=i;
      break;
    }
  }
}
```



```
for(i=size-1;i>=index;i--)  
    a[i+1]=a[i];  
size++;  
a[index]=newNumber;
```

0	1	2	3	4	5	6	7	..
10	20	30	40	50	60			

4.ต้องเลื่อน ตั้งแต่ตัวที่ 3 - 5 หรือ
ตั้งแต่ pos+1 ถึง size-1 ไป
ตำแหน่งที่ 4-6

5. ทำการแทรก



2) Delete

0	1	2	3	4	5	6	7
10	20	30	35	40	50	60	

1.หาตำแหน่งที่ต้องการลบใส่ตัวแปร index

2.ขยับข้อมูลตั้งแต่ลำดับ 4-6 คือ index+1 ถึง size – 1

เลื่อนมาด้านหน้า

3.ลดขนาด size

$$\begin{aligned}\frac{1+2+\dots+n}{n} &= \frac{\frac{n(n+1)}{n}}{n} = \frac{1}{2}(n^2+n) \\ &= \frac{n^2}{n} = O(n)\end{aligned}$$



การบ้าน

1.จงเขียนโปรแกรมโดยใช้ Array ขนาด 10 ช่อง สร้างเป็น list โดยมีการทำงานตาม menu ดังด้านล่าง

หมายเหตุ ให้ทดลอง insert 8 5 1 20 6 14 และลบ 8 20 1

=====Menu=====

+	1) Insert	+
+	2) Delete	+
+	3) Print	+
+	4) Exit	+

=====

Please choose >

ถ้าเลือกข้อ 1

Enter : 8

Output = 8 จากนั้นกลับไปเมนู

ถ้าเลือกข้อ 1

Enter : 5

Output = 5 8 จากนั้นกลับไปเมนู

ถ้าเลือกข้อ 2

Delete : 8

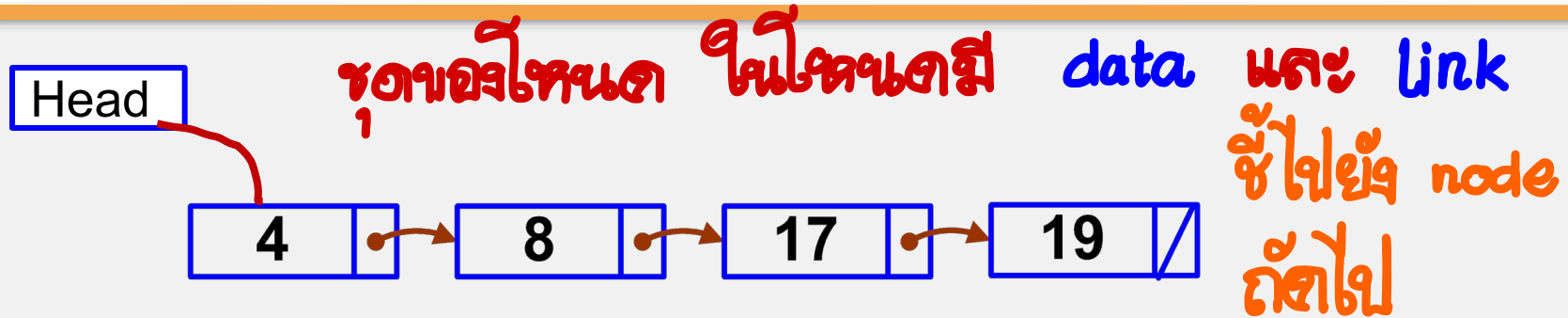
Output = 5 จากนั้นกลับไปเมนู



ถ้าเลือกข้อ 3 สมมติว่ามีข้อมูล 10 20 30 40 50 60
จะแสดงข้อมูลดังนี้

Print : 10 20 30 40 50 60
Print first half : 10 20 30
Print second half : 40 50 60

หรือ
Print : 10 20 30 40 50
Print first half : 10 20
Print second half : 30 40 50



Avoid the linear cost of insertion and deletion of array.

The linked list consists of a series of nodes, which are not necessary adjacent in memory. Each node contains the element and a link to a node containing its successor. We call this the next link. The last cell’s next link points to NULL.

last node = null

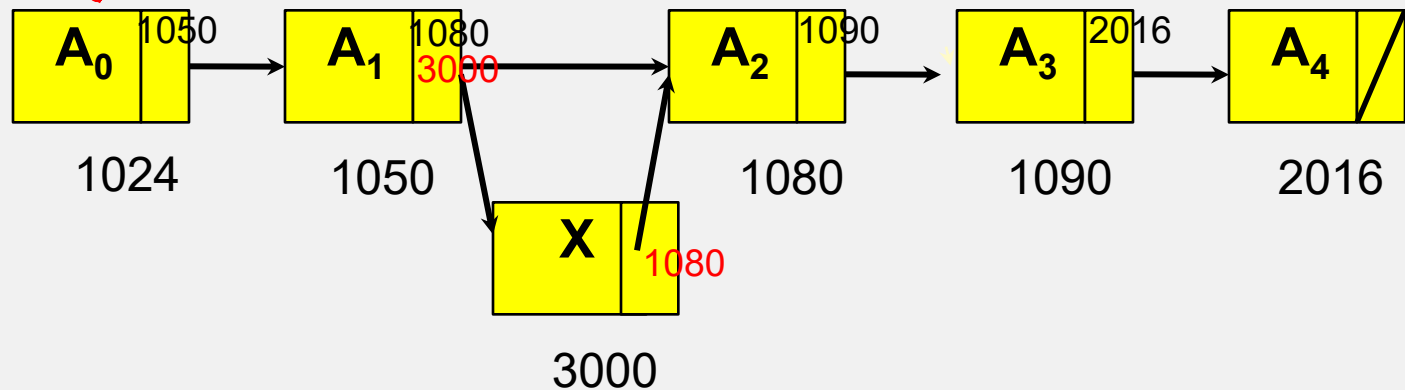
2.3.3 Linked List

Insert

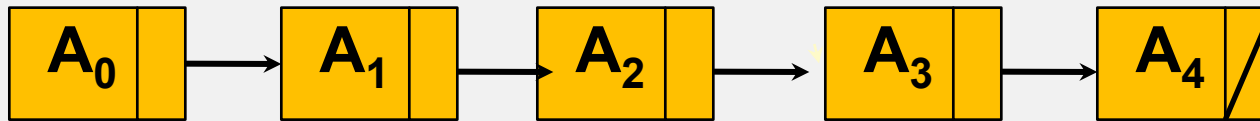
Head



2000

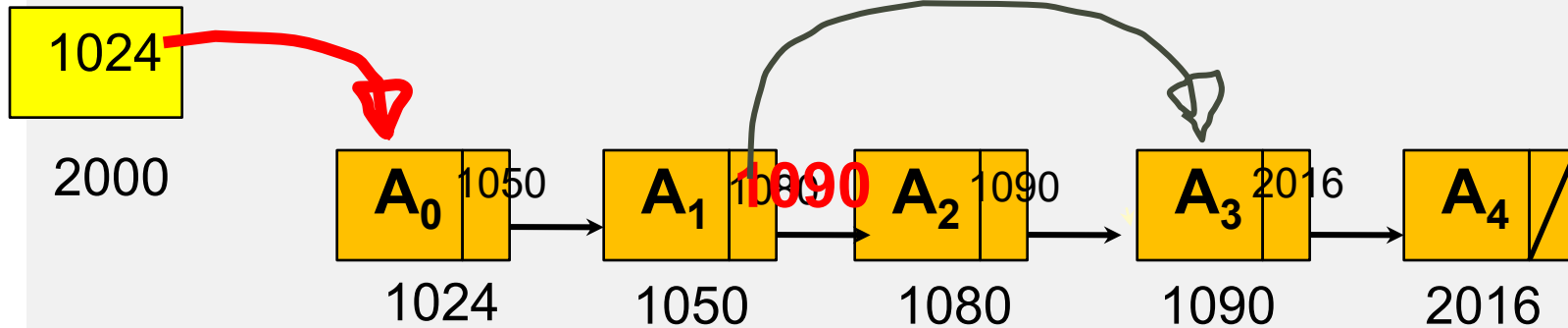


Delete



Delete

Head





ข้อแตกต่างระหว่าง Array list และ Linked list

1. การประกาศตัวแปร

Array list จะต้องประกาศตัวแปรก่อน จึงต้องคาดคะเนจำนวนข้อมูลไว้ ว่า list จะมีจำนวนกี่ตัว

Linked list ไม่จำเป็นจะต้องประกาศตัวแปรก่อน สามารถสร้าง node ขณะที่ run โปรแกรมได้

2. การ insert และ delete

Array list ทำได้ยากกว่า เพราะโครงสร้างไม่เหมาะสม
Linked list สามารถโปรแกรมได้ง่ายกว่ามาก

ข้อแตกต่างระหว่าง Array list และ Linked list(ต่อ)

3. การเขียนโปรแกรม

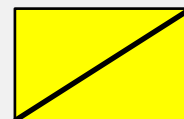
Array list เขียนโปรแกรมโดยใช้การวน loop

Linked list เขียนโปรแกรมต้องใช้ pointer

1 การ Insert แยกกรณี

1. กรณีที่ไม่มีข้อมูล

head



2000

head



2000



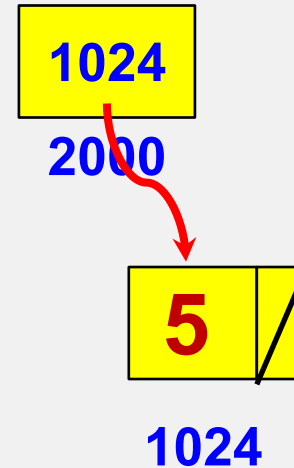
1024



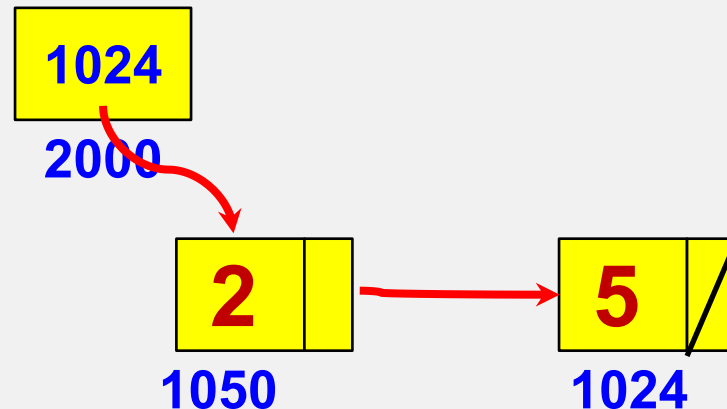
2. กรณีที่มีข้อมูล

- insert หน้าที่สุด
- insert ตรงกลาง
- insert ท้าย

head



head



Insert linked list

กรณี head ไม่มีข้อมูล

```
{ if(head==NULL)
```

สร้างโหนด

ทำการ insert และให้ head ชีโหนดนี้

```
}
```

```
else ( head มีข้อมูล)
```

```
{
```

insert หน้า head?

insert และให้ head มาชี้

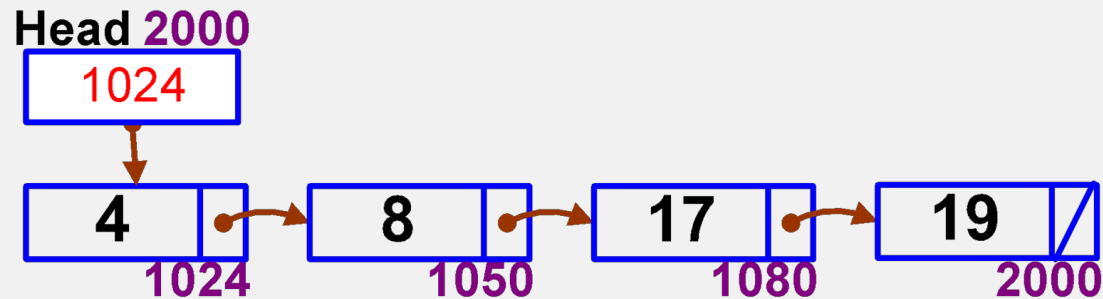
else (insert หลัง head คือตรงกลางหรือท้าย)

```
{ - หาดำแหน่งที่จะ insert
```

```
- ทำการ insert
```

```
}
```

```
}
```



```

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



//กำหนด structure ของ linked list

head->next->next = 1080

head->next->next->value = 17

&head = 2000

head เก็บค่า 1024

head->value = 4

head->next = 1050



```
int menu()
{
    int choose;
    cout << "=====Menu =====\n";
    cout << " 1) Insert list\n";
    cout << " 2) Delete list\n";
    cout << " 3) Print list\n";
    cout << " 4) Exit\n";
    cout << " Please choose > ";
    cin >> choose;
    return choose;
}
```

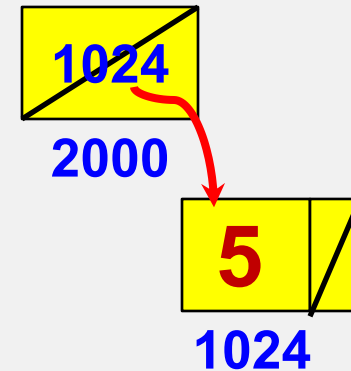



```
struct record *insert(struct record *head,int data)
```

```
1 { struct record *node,*p;
2   if ( head == NULL )
3   { head=new struct record;
4     head-> value = data;
5     head-> next = NULL;
6   }
7   return head;
```

return &head; 1024 2000

head



5

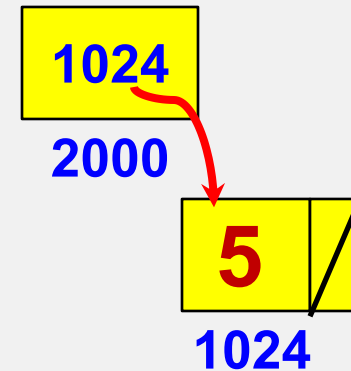


```
struct record *insert(struct record *head,int data)
```

```
1 { struct record *node,*p;  
2   if ( head == NULL )  
3   { head=new struct record;  
4     head-> value = data;  
5     head-> next = NULL;  
6   }
```

2

head



```

7 else    /**head !=NULL **/
8 {      node=new struct record;
9        node-> value = data;
10      if( data < head->value) ***
11      {  node->next = head;
12        head=node;
13      }
14      }
15return head;

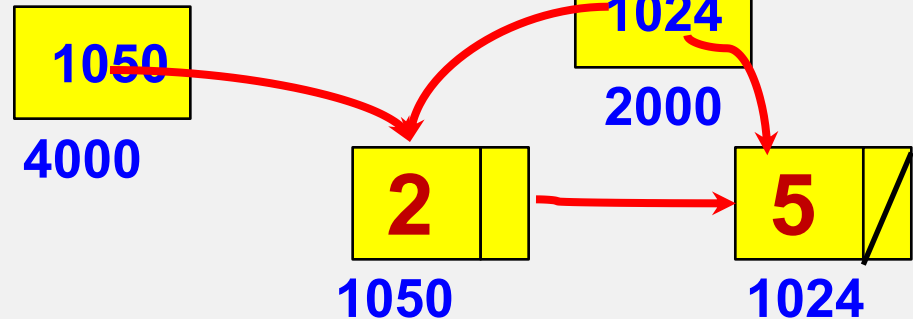
```

2. กรณีมีข้อมูลอยู่
แล้ว

-Insert ด้านหน้า

node

head



```

struct record *insert(struct record *head,int data)

```

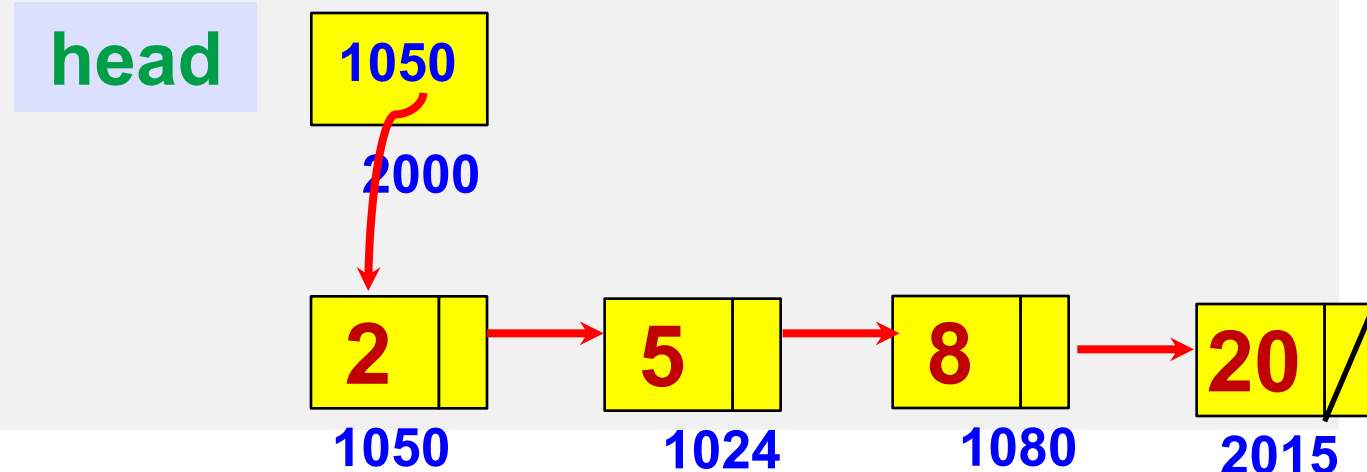
```

1 {   struct record *node,*p;
2     if ( head == NULL )
3     { head=new struct record;
4         head-> value = data;
5         head-> next = NULL;
6     }

```

3. กรณีแทรกกลาง
หรือท้าย

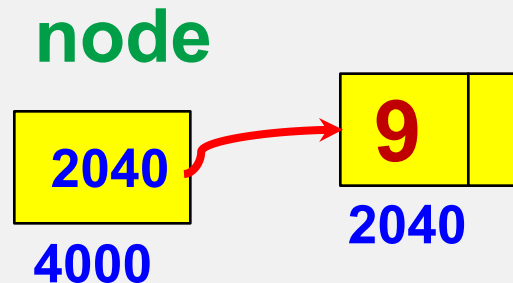
9





```
7 else    /**head !=NULL **/  
8 {      node=new struct record;  
9        node-> value = data;  
10       if( data < head->value)  
11       {  
12          แทรกหน้า list กรณี 2 เรียนแล้ว ❤️  
13           ...  
        }  
        else  
        {     แทรกกลาง/ท้าย
```

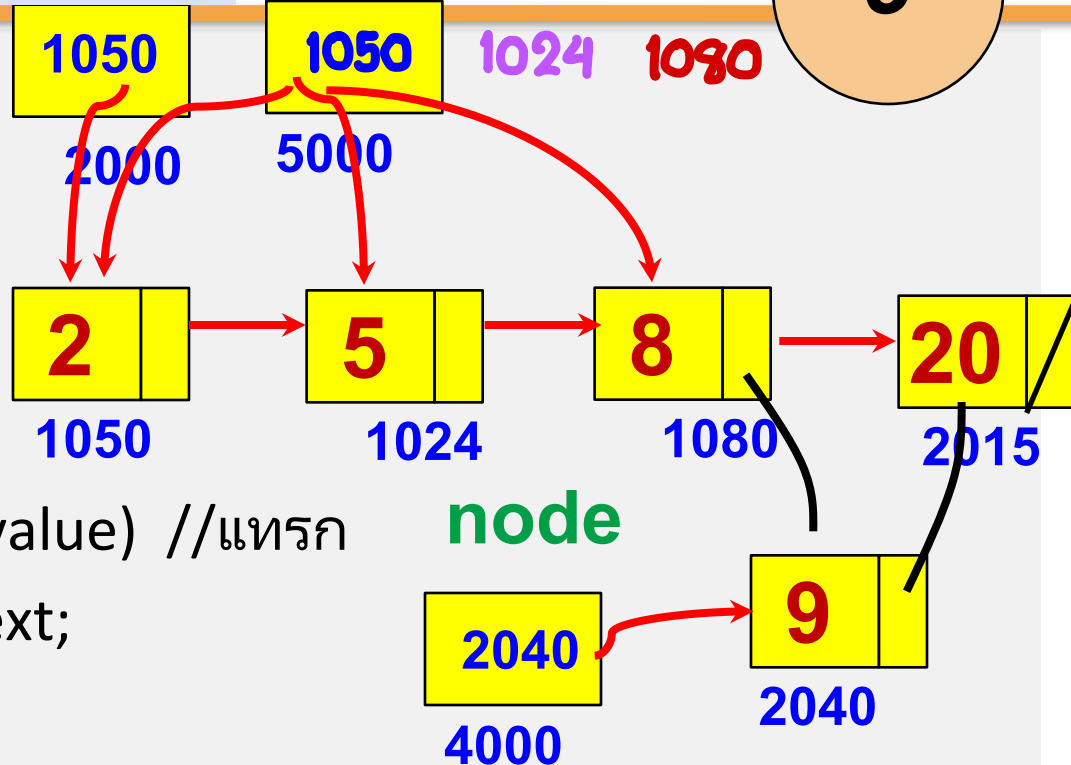
3. กรณีแทรกกลาง



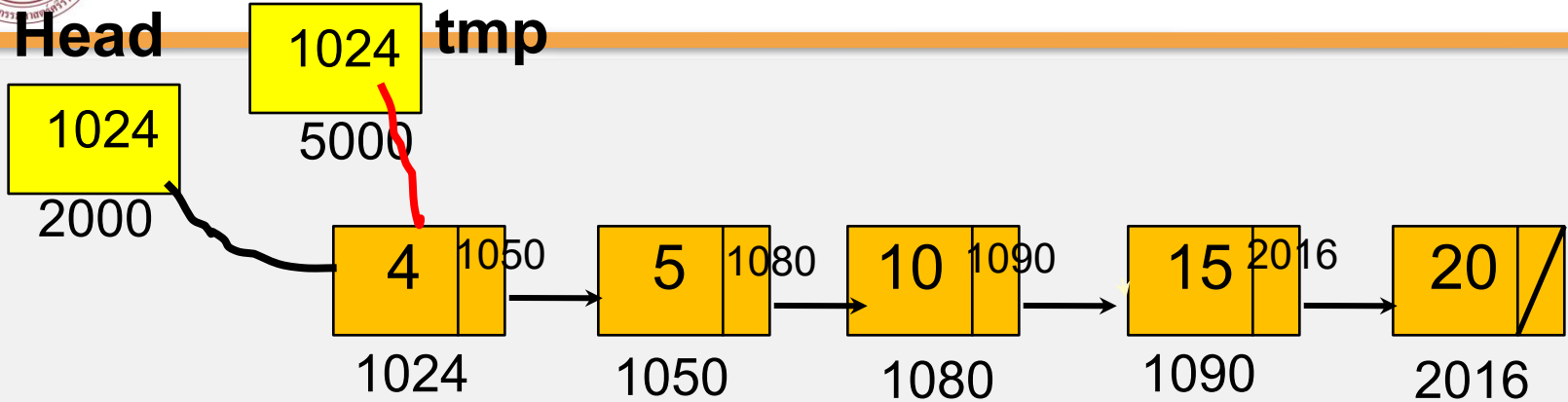
head

p

9



14 else
15 { p=head;
16 while(p !=NULL)
17 {
18 if(data < p->next->value) //แทรก
19 { node->next=p->next;
20 p->next = node;
21 break;
22 }
23 else p=p->next;
24 } //end while } //end else /* end else head !=NULL */
25 return head; }



```
void print(struct record *head)
{
    cout << "\nPrint Listed : \n";
    struct record *tmp;
    tmp=head;
    while(tmp!=NULL)
    {
        cout << tmp->value << " ";
        tmp=tmp->next;
    }
}
```