2.0 Mathematics Review

1) Exponents

$X^A X^B = X^{A+B}$

$\frac{X^A}{X^B} = X^{A-B}$

$(X^A)^B = X^{AB}$

$X^N + X^N = 2X^N \neq X^{2N}$

$X^N + X^N = 2^{N+1}$

2) Logarithms

$X^A = B \rightarrow \log_X B = A$

$\log_A B = \frac{\log_c B}{\log_c A}$

3) Series

Exponent

$2^0 + 2^1 + \ldots + 2^N$

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

$A^0 + A^1 + \ldots + A^N$

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1} - 1}{A - 1}$$

3) Series

Exponent

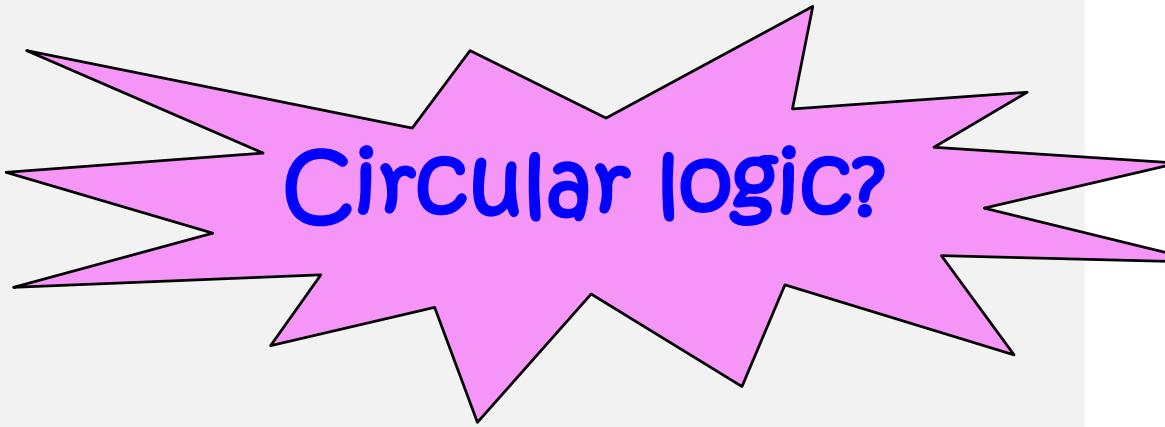
$1 + 2 + 3 + \ldots + N = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

$1^2 + 2^2 + 3^2 \ldots + N^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1}{6}$

2.1 A Brief Introduction to Recursion :

Mathematical function

1. C=2(F-32)/9



2. f(x) = x*f(x-1)
   f(1) = 1

3. $f(x) = 2f(x-1)+x^2$
   f(0)=0 , x nonnegative integer

Example 1

```
#include <stdio.h>
int fact(int x)
{   if(x <= 0)
        return 1;
    else
        return x* fact(x-1);
}
```

```
int main()
{    int ans;
     ans = fact(3);
     cout << ans;
}
```

Example 2 Factorial

```
int main()
{    int ans;
     ans = f3(3);
     cout << ans;
}
```

```
int f3(int x)
{   if(x <= 0)
        return 1;
    else
        return x* 2
}
```

```
int f2(int x)
{   if(x <= 0)
        return 1;
    else
        return x* 1;
}
```

```
int f1(int x)
{   if(x <= 1)
        return 1;
    else
        return x* f(x-1);
}
```

Exercise 1

จงเขียนโปรแกรมหา ค่า 1*2*3*...*n โดยใช้วิธี recursive

```
int main()
{  int n=4;
   cout << recur(n);
}
```

Example 3

```
int bad(int n)
{   if (n==0)
       return 0;
    else
       return bad(n/3 + 1+ n – 1);
}
```

Example 4

```
void printout(int n)
{   if( n >=10 )
       printout(n/10);
    cout << n%10;
}
```

2.2 Algorithm Analysis

Definition : An algorithm is a clearly specified set of simple instructions to be followed to solve a problem.

- correct
- time or space

1) Mathematical Background

Definition : T(N)=O(f(N)) if there are positive constants c and n_0 such that $T(N) \leq cf(N)$ when $N \geq n_0$.

- Give two functions
- we compare their relative rates of growth.

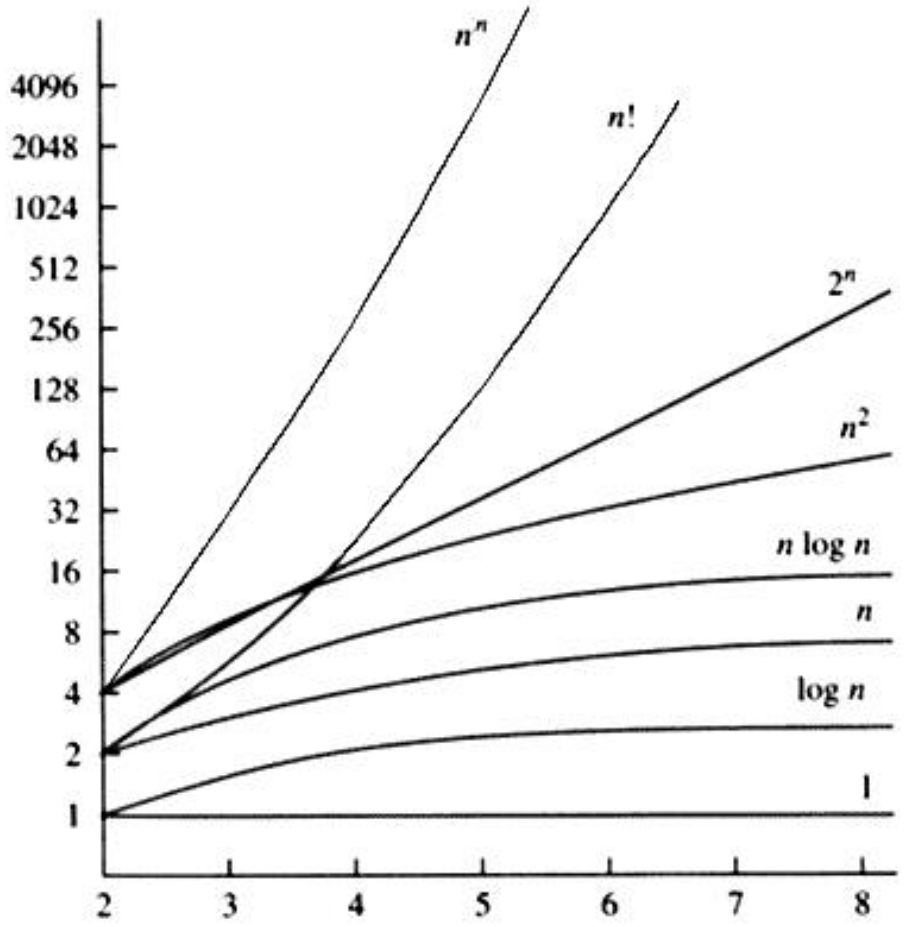
Although 1000N is larger than N^2 for small value of N, N^2 grow at a faster rate, and thus N^2 will eventually be the larger function.

T(N)=O(f(N)) if there are positive constants c and n_0 such that T(N) ≤ cf(N) when N ≥ n_0.

- T(N)=1000N
- f(n)= N^2
- N_0=1000
- c=1

We can say that 1000N = O(N^2)

Function	Name
C	Constant
logN	Logarithmic
log^2N	Log-squared
N	Linear
NlogN	
N^2	Quadratic
N^3	Cubic
2^N	Exponential



3) Running time Calculations

```
int sum(int n)
{       int partialSum;
        partialSum=0;
        for(int i=1; i<=n; i++)
                partialSum += i*i*i;
        return partialSum;
}
```

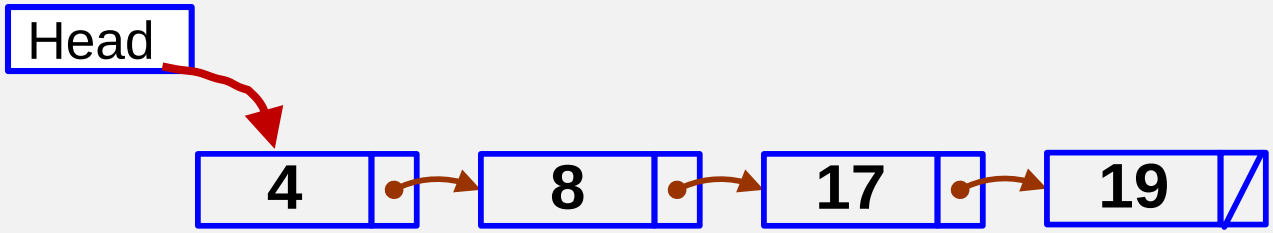
Simple and basic data structures

- List
- Stack
- Queue

2.3 The List

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

For any list except the empty list, we say that A_{i+1} follows (or succeeds) A_i ($i<N$) and that A_{i-1} 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

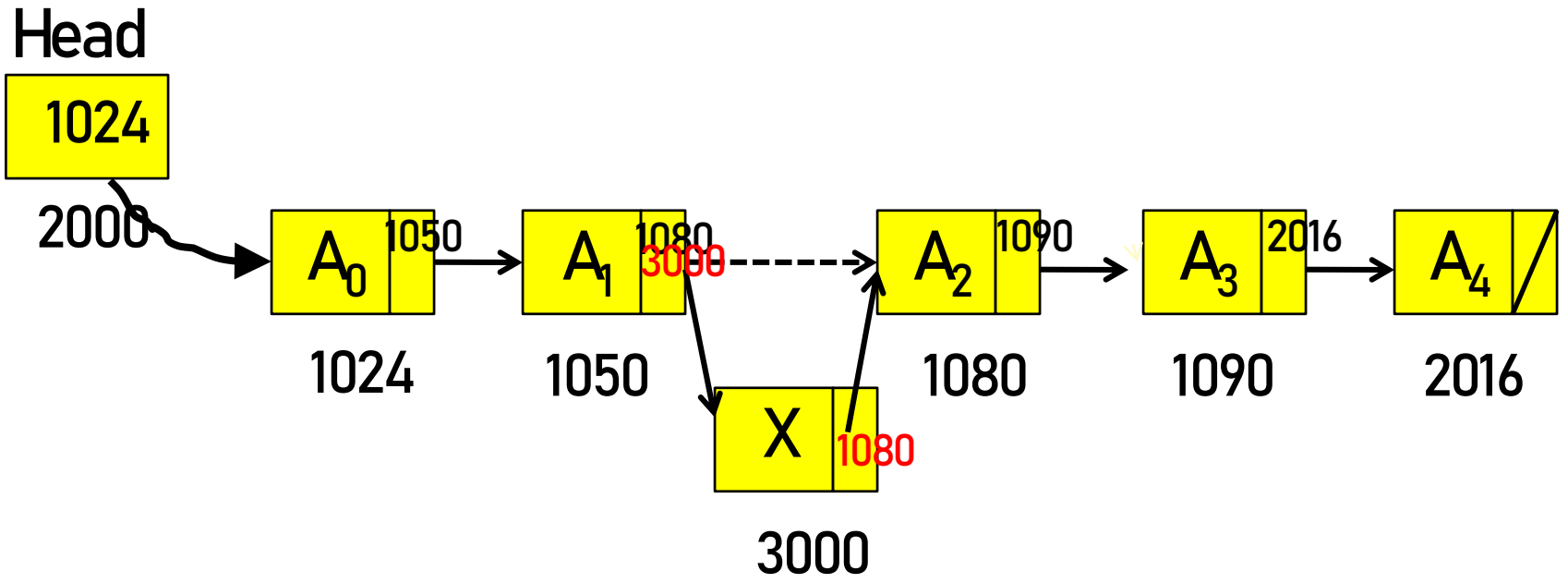


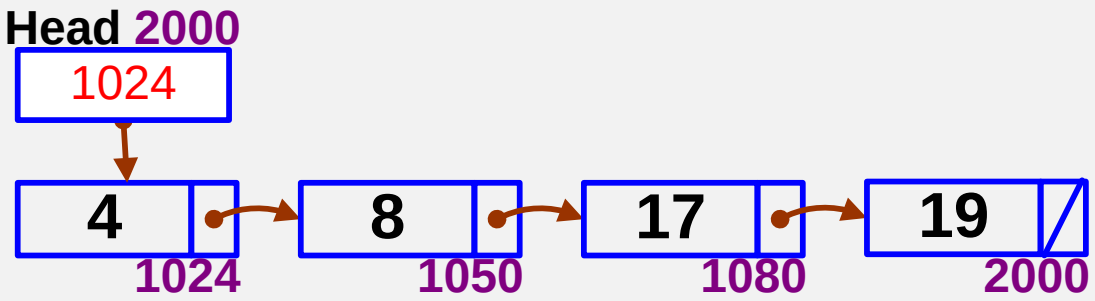
Avoid the linear cost of insertion and deletion

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.

2.3.1 List Operation

- insert
- remove
- printList
- makeEmpty
- find


Head
1024
2000
A0
1050
A1
1080
3000
A2
1090
A3
2016
A4
1024
1050
1080
1090
2016
X
1080
3000




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

```
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;
}
```

1 การ Insert แยกกรณี

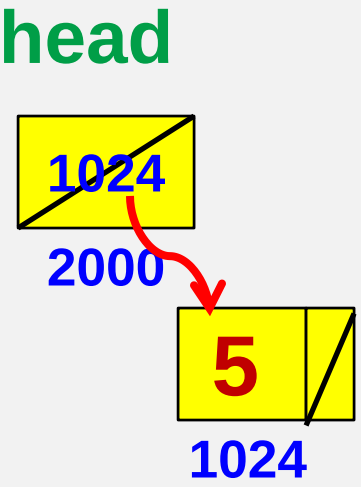
```
struct record *insert(struct record *head,int data)
{
        1. Insert กรณี list ว่าง

        2. กรณีที่มีข้อมูล
                - insert หน้าสุด
                - insert ตรงกลาง
                - insert ท้าย

         return head;
}
```

8

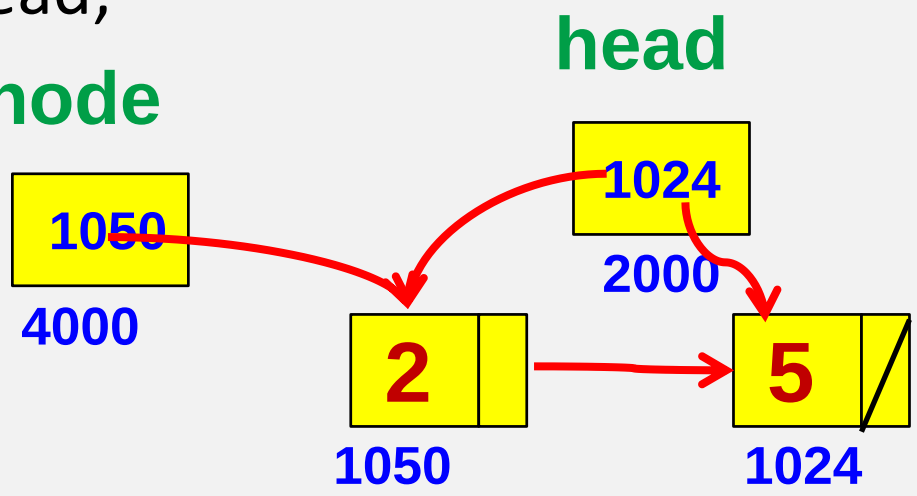
```
struct record *insert(struct record *head,int data)
1 {     struct record *node,*p;
2       if ( head == NULL )
3       {   head=new struct record;
4           head-> value = data;
5           head-> next = NULL;
6       }
```

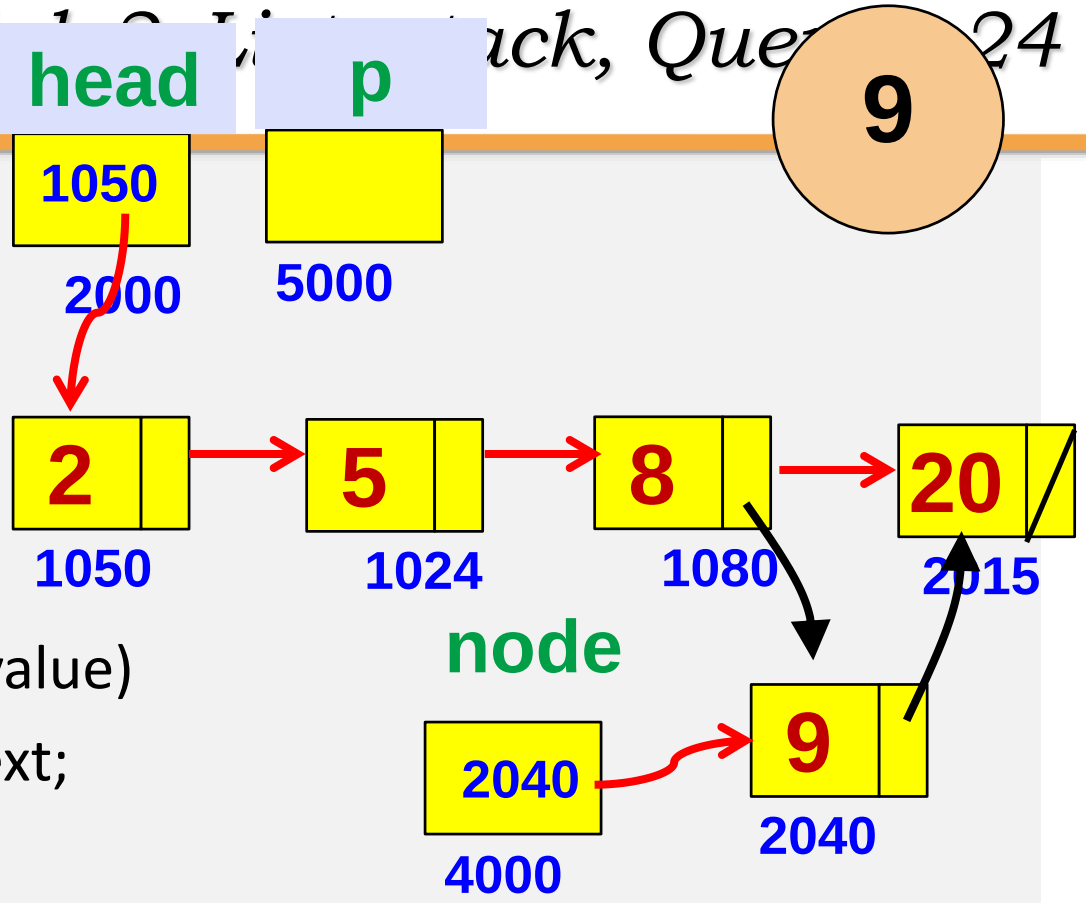


```
7 else      /**head !=NULL **/
8 {     node=new struct record;
9       node-> value = data;
10      if( data < head->value)
11      {    node->next = head;
12           head=node;
13      }
```

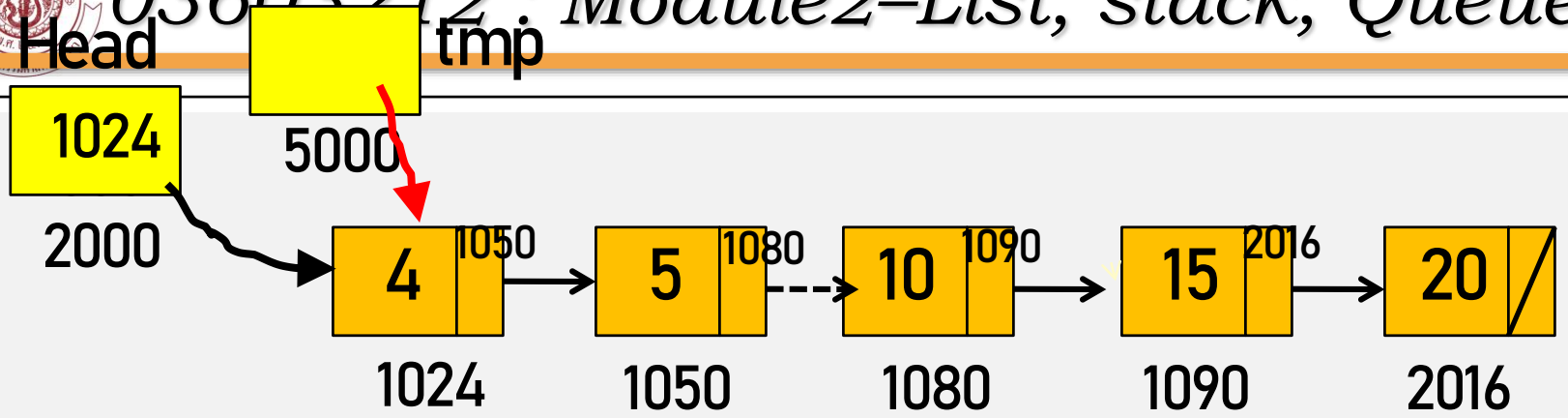
2

2. กรณีมีข้อมูลอยู่แล้ว
-Insert ด้านหน้า



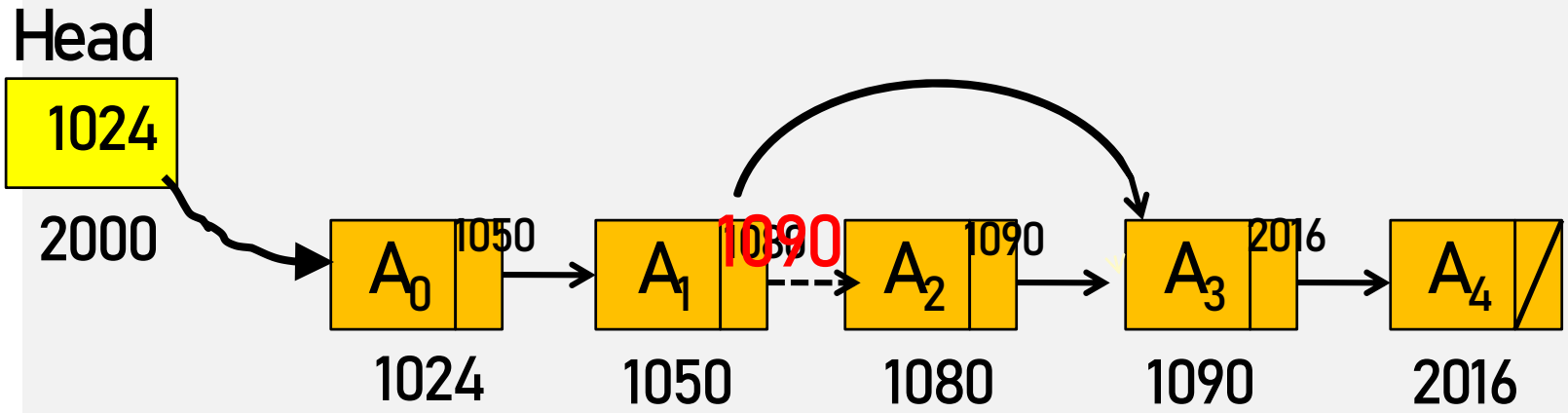



```
14 else
15 {   p=head;
16     while( p !=NULL)
 17   {
18         if( data < p->next->value)
19        {   node->next=p->next;
20            p->next = node;
21           break;
22        }
23        else  p=p->next;
24     }//end while      }  //end else    }/* end else head !=NULL */
25   return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



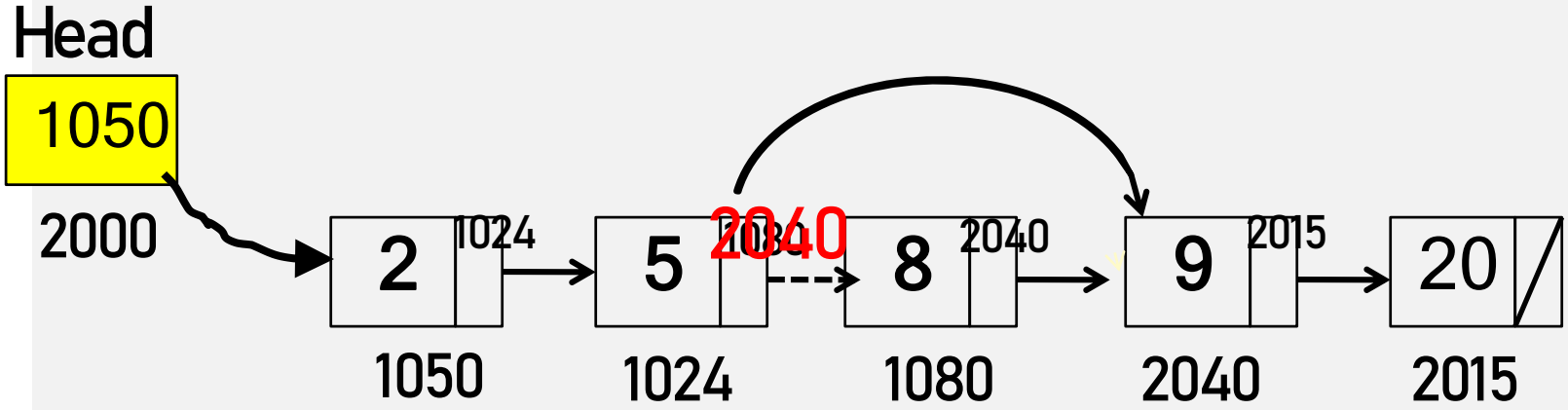

```
void print(struct record *head)
{       cout << "\nPrint Listed : \n";
        struct record *tmp;
        tmp=head
        while(tmp!=NULL)
        {       cout << tmp->value << " ";
                tmp=tmp->next;
        }
}
```

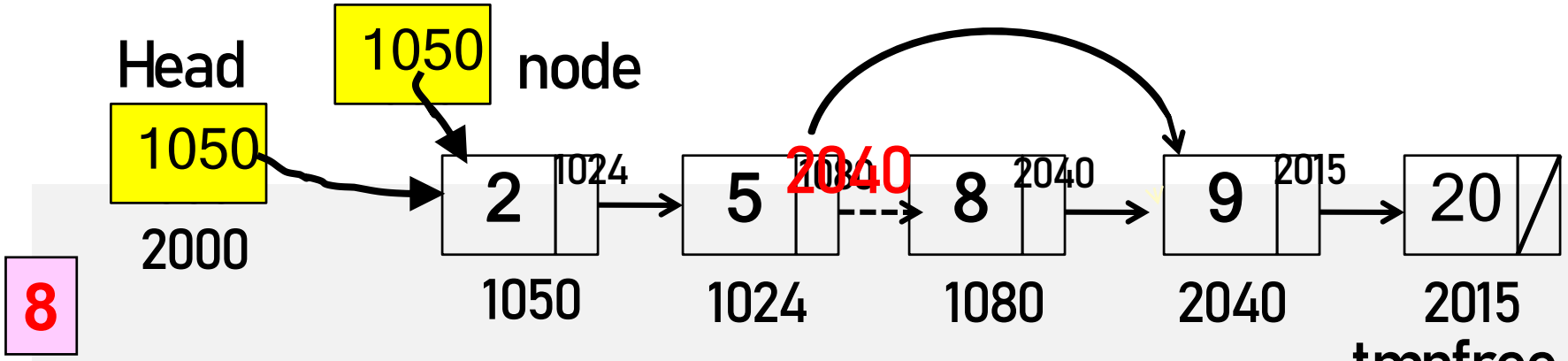
Delete



ต้องการลบก้อน A2

Delete

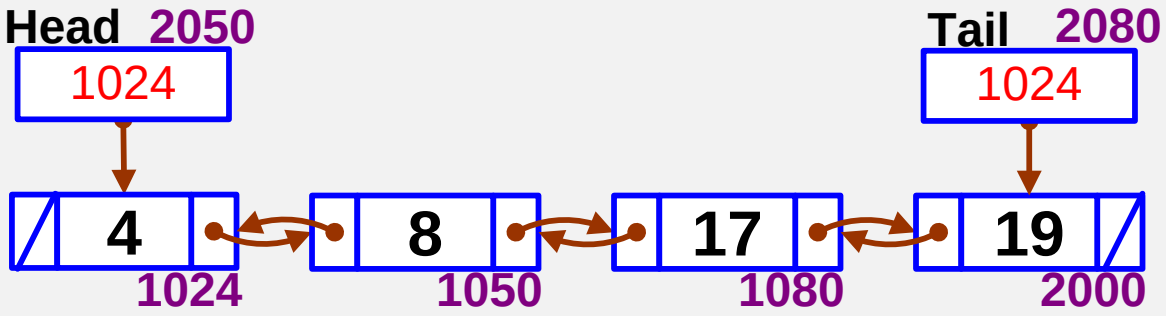

Head
1050
2000
2
1024
5
2040
8
2040
9
2015
20
1050
1024
1080
2040
2015




```
struct record *delete(struct record *head,int data)
{   struct record *node,*tmpfree;
    node = head;
    while(node)
    {   if( data == node->next->value )
        {   tmpfree = node->next;
            node->next = node->next->next;
            delete(tmpfree);
            break;
        }
        else
            node=node->next;
    } /* end while */
    return node;
}
```

2.4 Doubly Linked Lists

The link list that add extra field to the data structure, containing a pointer to the previous cell.




```
struct record
{    int data;
     struct record *next;
     struct record *prev;
};
```

1 การ Insert แยกกรณี

```
struct record *tail=NULL;
struct record *insert(struct record *head,int data)
{
```

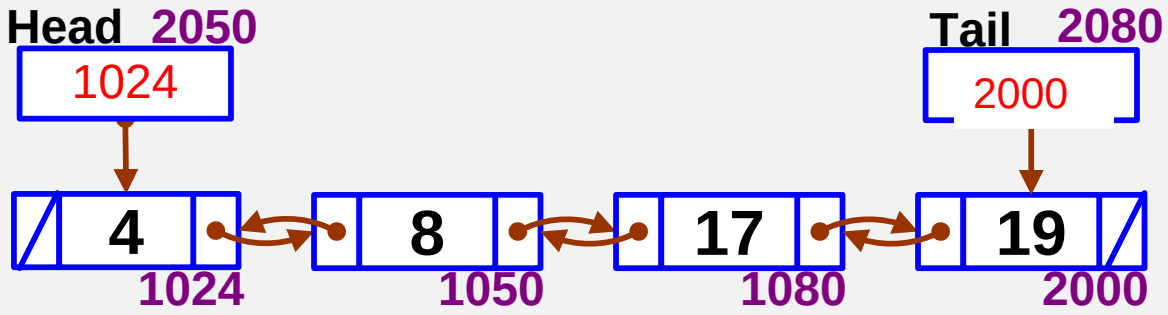
1. Insert กรณี list ว่าง

2. กรณีที่มีข้อมูล
   - insert หน้าสุด
   - insert ตรงกลาง
   - insert ท้าย

```
    return head;
}
```

2.4.1 Insertion (Doubly Linklist)

- Insert while no data in list
- Insert first
- Insert last
- Insert middle



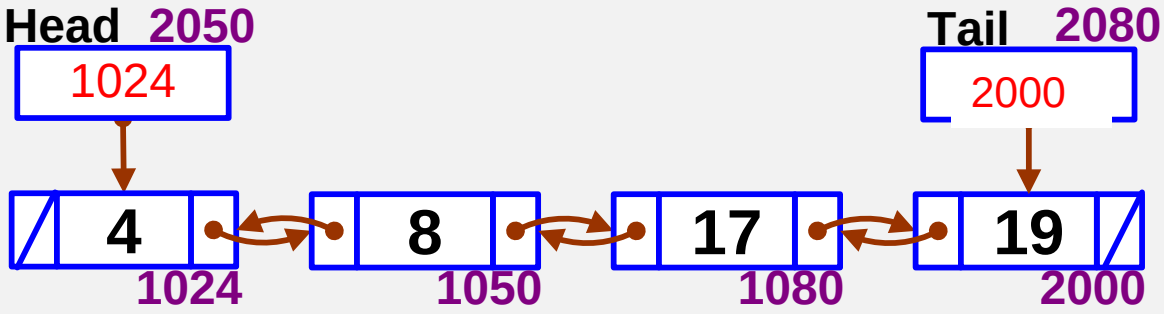
```
struct node{
        int value;
        struct node *prev;
        struct node *next;
};
```

```
struct node  *tail=NULL;
void insert(struct node *head, int data)
{
    - Insert while no data in list
    - Insert first
    - Insert last
    - Insert middle
   return head;
}
```

```
if( (head== NULL)
      {   head=new struct rec;
          head->value= data;
          head->next=NULL;
          head->prev=NULL;
          tail=head;
```

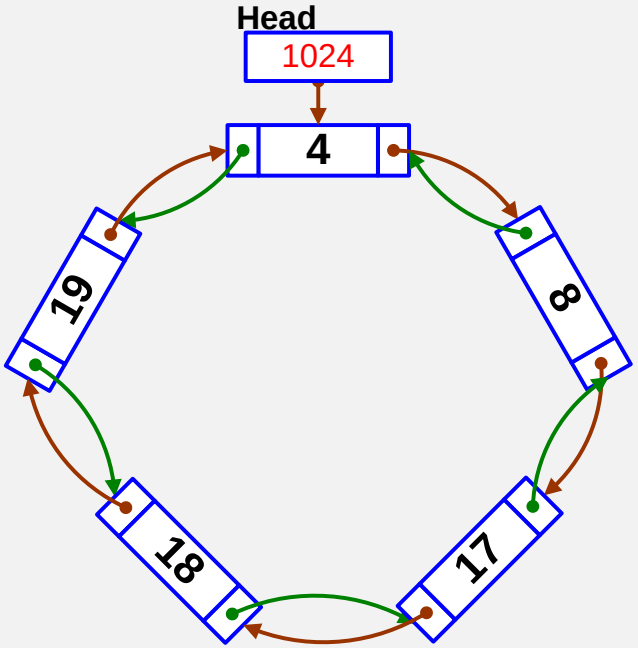
```
else
    {   tmp=head;
        if (temp->value >= data )
        {
```

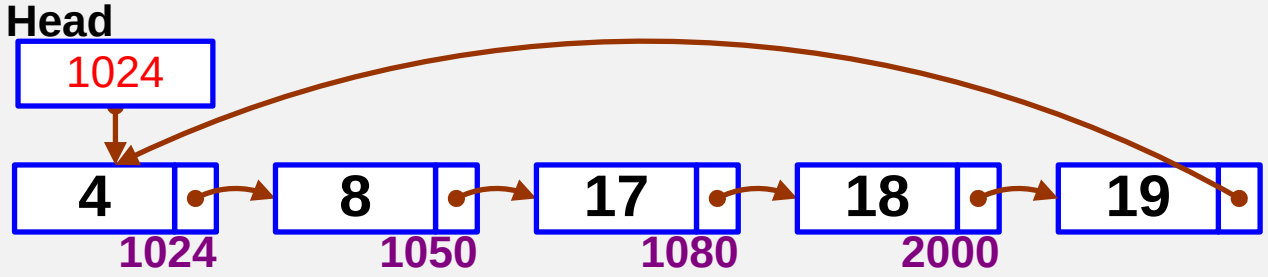
2.4.2 Deletion (Doubly Linklist)



2.5 Circularly linked lists

A popular convention is to have the last cell keep pointer back to the first. This can be done with or without a header (If the header present, the last cell point to it.)



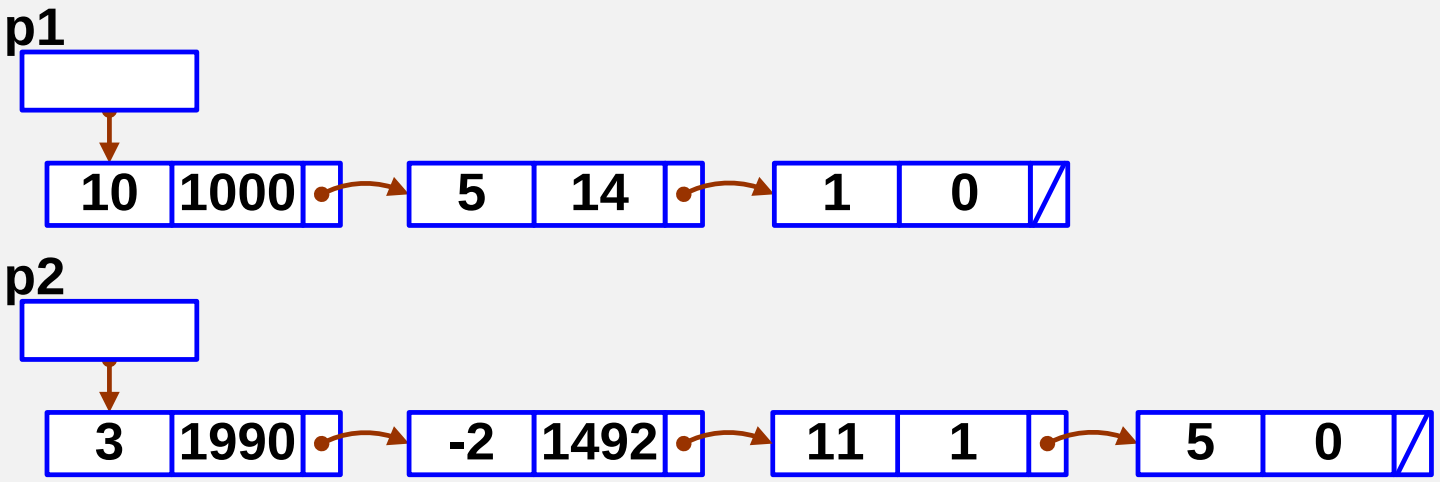


2.6 Examples

2.6.1 The polynomial ADT

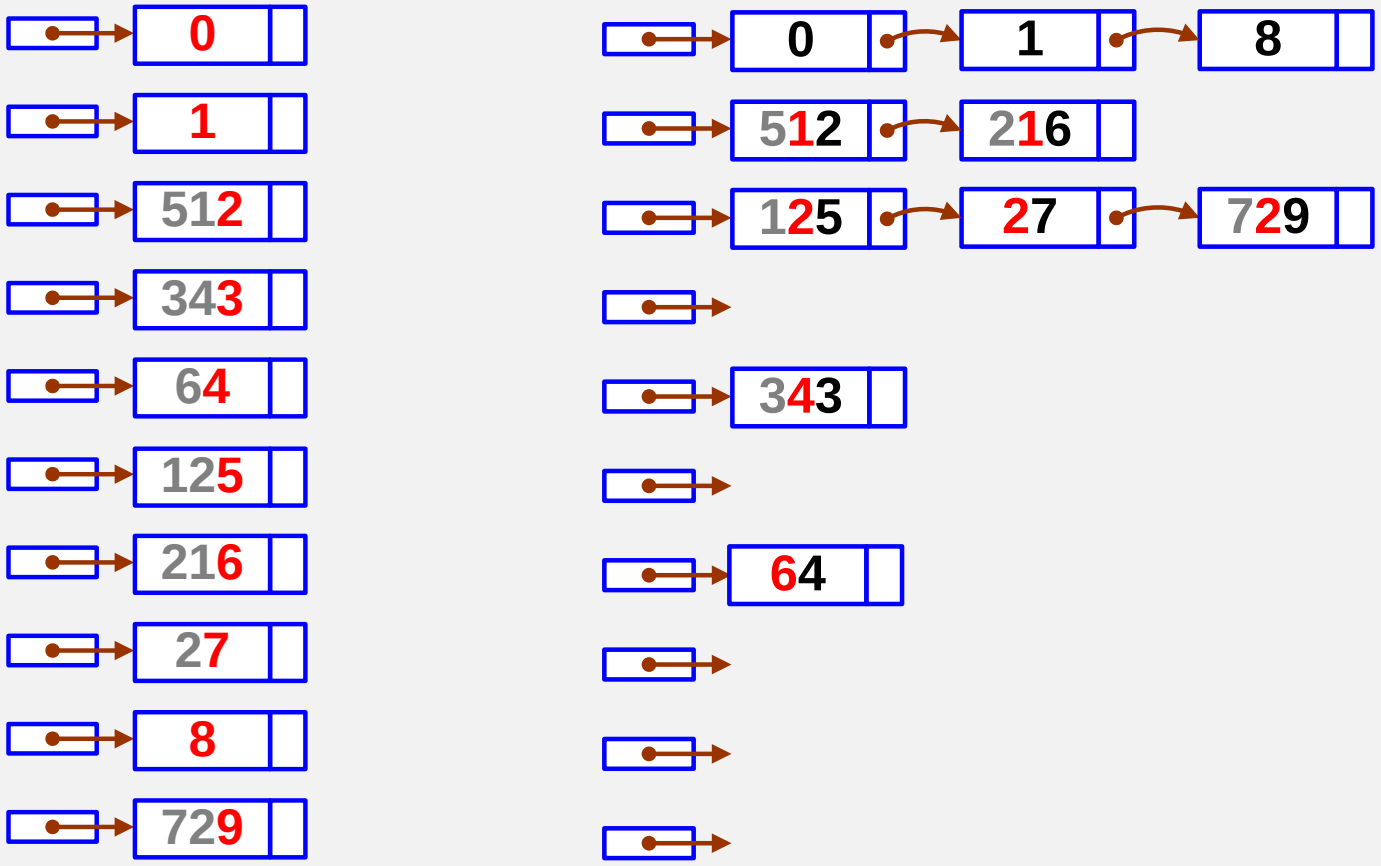
p1(x) = 10x^1000 + 5x^14 + 1

p2(x) = 3x^1990 - 2x^1492 + 11x + 5



2.6.2 Radix Sort

Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125



2.6.3 Multilists