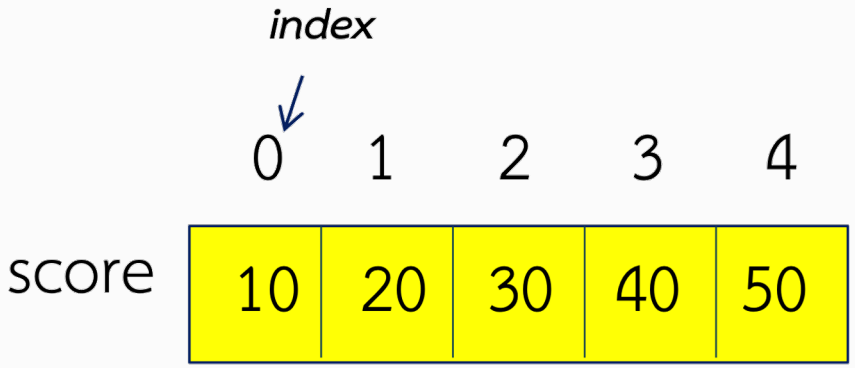
Labs 10: Singly Linked List

Lab Documentation

Array

- **Array** เป็นโครงสร้างข้อมูลที่จัดเก็บข้อมูลเป็นลำดับแบบที่มีขนาดคงที่ของชนิดข้อมูลเดียวกัน โดยทำการจองพื้นที่ในหน่วยความจำต่อเนื่องกัน

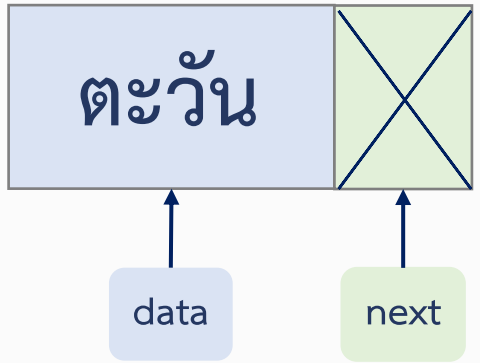


- **Static Memory Allocation**
  (การจัดสรรหน่วยความจำแบบคงที่)
  - เมื่อมีการประมวลผล เนื้อที่เหล่านี้ไม่สามารถขยายหรือลดลงได้
  - ไม่มีความยืดหยุ่น

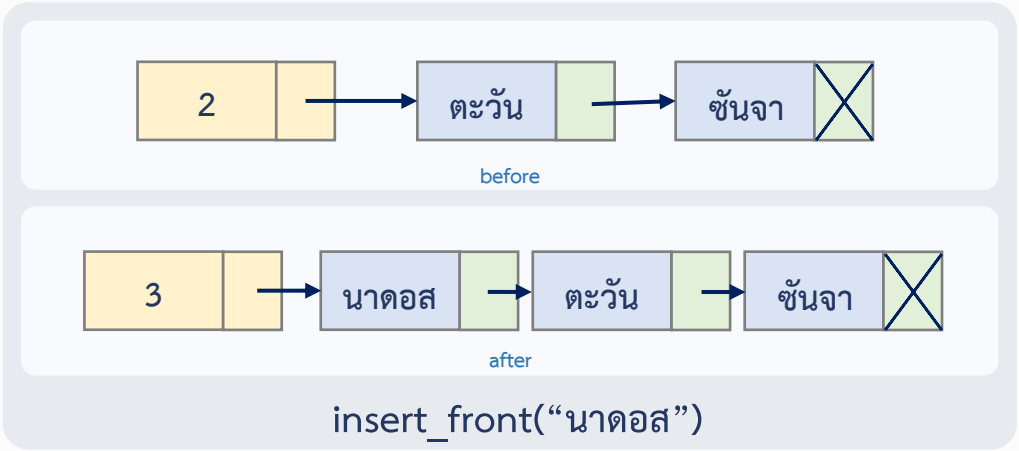
```
int score[] = {10, 20, 30, 40, 50};
score[1] = -20;
printf("%d", score[2]); // Output is 30
```

Linked List

- **Linked List** เป็นโครงสร้างข้อมูลที่องค์ประกอบต่างๆ จะถูกเก็บไว้ในโหนด แต่ละโหนดประกอบด้วยส่วนประกอบหลักสองส่วน
  - **Data:** The actual value stored in the node.
  - **Pointer:** A reference to the next node.

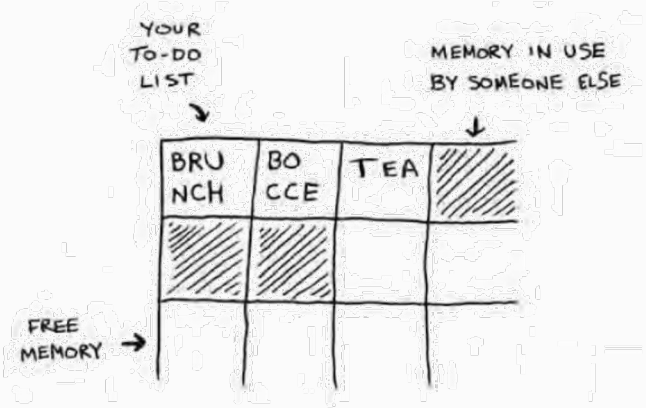


- **Dynamic Memory Allocation (การจัดสรรหน่วยความจำแบบไดนามิก)**
  - สามารถสร้างตัวแปรขึ้นได้ทุกครั้งที่ต้องการใช้ และสามารถทำลายลงเมื่อไม่ต้องการ
  - มีความยืดหยุ่น

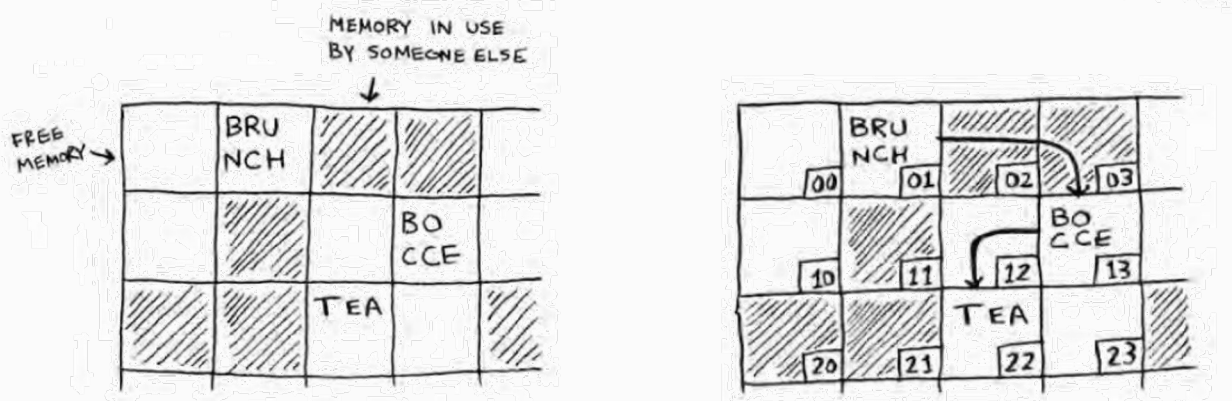


Array VS Linked List

คุณสมบัติ	Array	Linked List
ขนาด	คงที่	ไดนามิก (สามารถขยายหรือลดได้)
การจอง/จัดสรรหน่วยความจำ	ต่อเนื่อง	ไม่ต่อเนื่อง
การเข้าถึง	โดยตรง (ใช้ index)	ต่อเนื่อง (ใช้ pointer)



Array



Linked List

Linked List

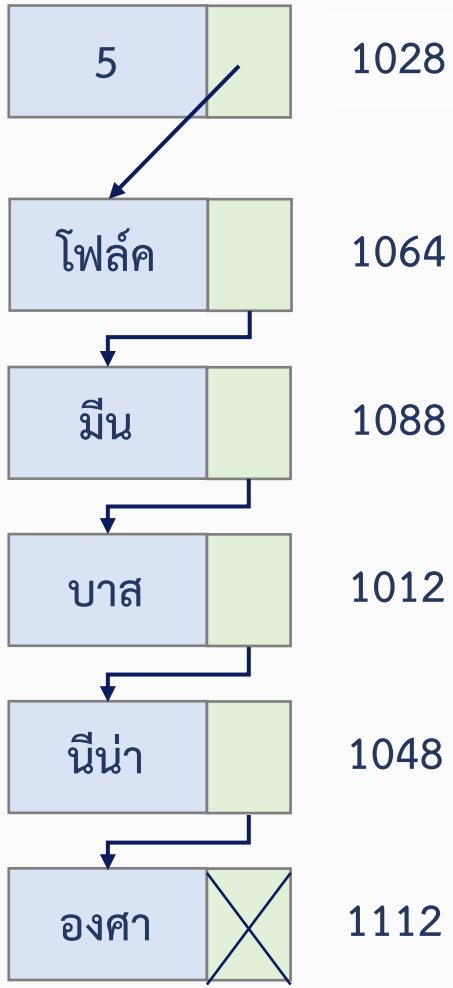
1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	data: บาส, next: 1048
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 5, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	data: นีน่า, next: 1112
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: 1012
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	data: องศา, next: null
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***

```
data = []
```



Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	. . . . .
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 0, head: null
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	. . . . .
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	. . . . .
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	. . . . .
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	. . . . .
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***

1028 (Head Node)

โฟล์ค	1064
มีน	1088
บาส	1012
นีน่า	1048
องศา	1112

```
data = []
```

สร้าง List ขึ้นมา

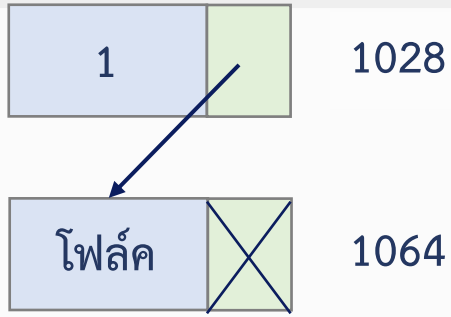
Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	. . . . .
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 1, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	. . . . .
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	data: โฟล์ค, next: null
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	. . . . .
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	. . . . .
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***



1028 (Head Node)

1064

มีน 1088

บาส 1012

นีน่า 1048

องศา 1112

```
data.append("โฟล์ค")
```

เพิ่มชื่อ “โฟล์ค”

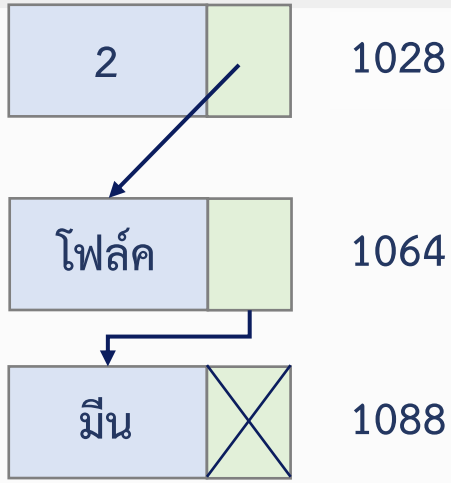
Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	. . . . .
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 2, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	. . . . .
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: null
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	. . . . .
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***




```
data.append("มีน")
```

เพิ่มชื่อ “มีน”

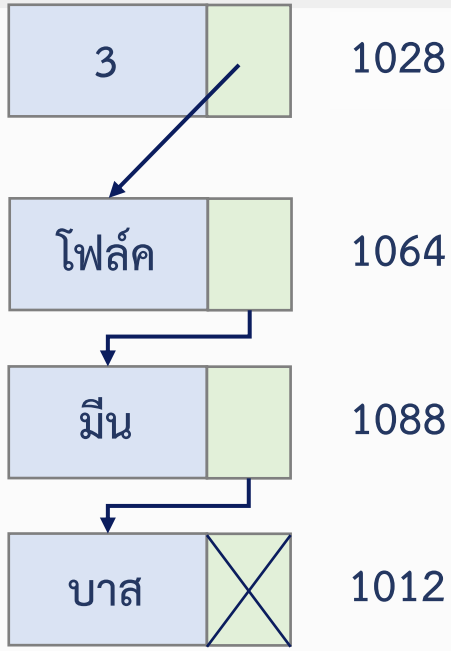
Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	data: บาส, next: null
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 3, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	. . . . .
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: 1012
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	. . . . .
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***




```
data.append("บาส")
```

เพิ่มชื่อ “บาส”

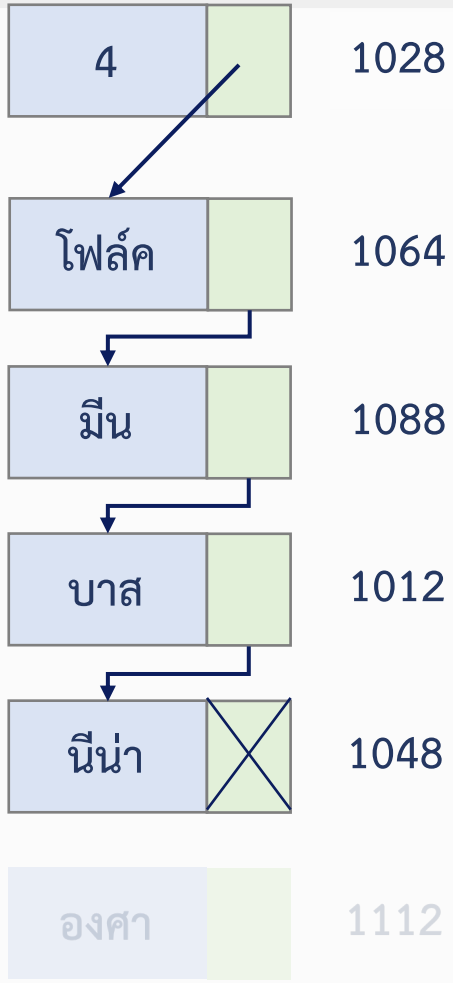
Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	data: บาส, next: 1048
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 4, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	data: นีน่า, next: null
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: 1012
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	. . . . .
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***




```
data.append("นีน่า")
```

เพิ่มชื่อ “นีน่า”

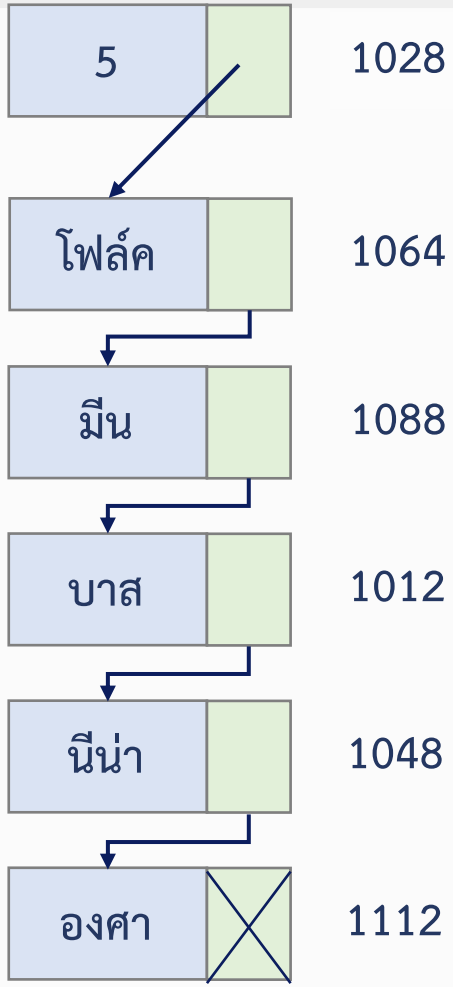
Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	data: บาส, next: 1048
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 5, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	data: นีน่า, next: 1112
1052	. . . . .
1056	. . . . .

data

1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: 1012
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	data: องศา, next: null
1116	. . . . .

นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***




```
data.append("องศา")
```

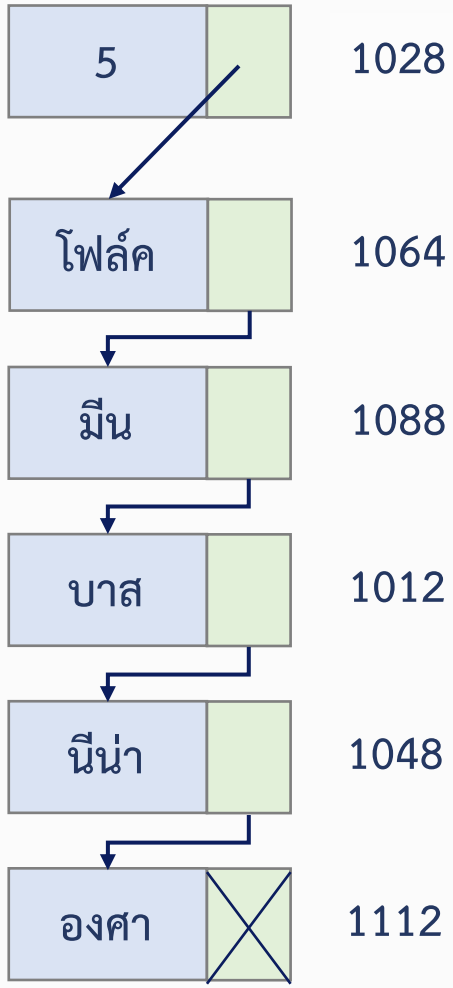
เพิ่มชื่อ “องศา”

Linked List

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	data: บาส, next: 1048
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 5, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	data: นีน่า, next: 1112
1052	. . . . .
1056	. . . . .

data

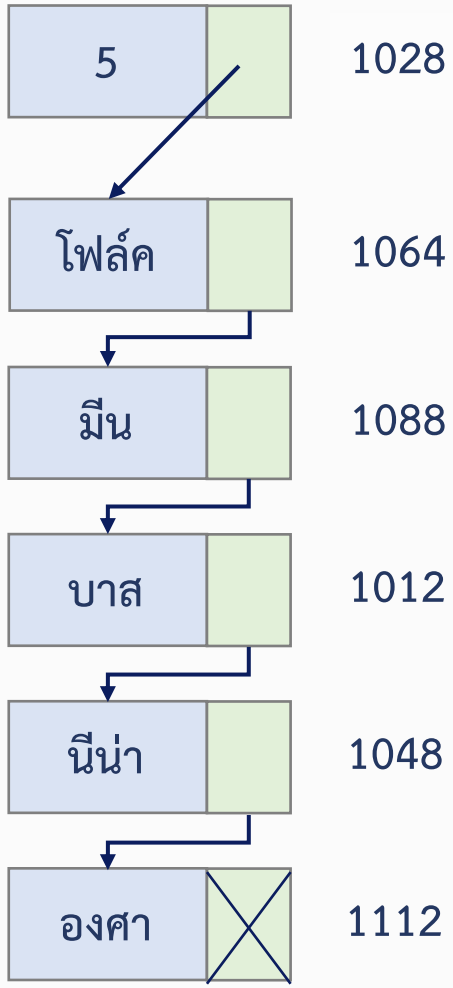
1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: 1012
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	data: องศา, next: null
1116	. . . . .



นี่คือการสมมุติ ขนาดของข้อมูลจริงไม่ใช่ตามตัวอย่างสมมุติ ***

Linked List

- ถ้าข้อมูลเยอะๆ เข้าถึงข้อมูลจะช้ามากๆ
- สามารถขยายตัวเองได้เรื่อยๆ

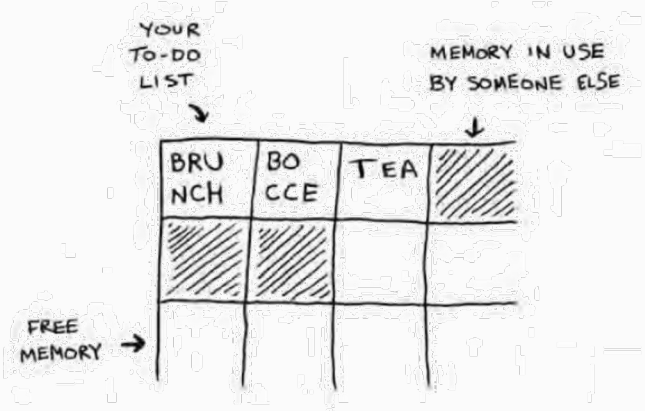


Array

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	66070199
1016	66070200
1020	66070207
1024	66070210
1028	66070212
1032	0
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	. . . . .
1052	. . . . .
1056	. . . . .

num

- Array เป็นโครงสร้างข้อมูลที่จัดเก็บข้อมูลเป็นลำดับแบบที่มีขนาดคงที่ของชนิดข้อมูลเดียวกัน
  - จองพื้นที่ในหน่วยความจำต่อเนื่องกัน
  - เนื้อที่เหล่านี้ไม่สามารถขยายหรือลดลงได้
  - ไม่มีความยืดหยุ่น



Array vs Linked List

num

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	66070199
1016	66070200
1020	66070207
1024	66070210
1028	66070212
1032	0
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	. . . . .
1052	. . . . .
1056	. . . . .

1000	. . . . .
1004	. . . . .
1008	. . . . .
1012	data: บาส, next: 1048
1016	. . . . .
1020	. . . . .
1024	. . . . .
1028	count: 5, head: 1064
1032	. . . . .
1036	. . . . .
1040	. . . . .
1044	. . . . .
1048	data: นีน่า, next: 1112
1052	. . . . .
1056	. . . . .

1060	. . . . .
1064	data: โฟล์ค, next: 1088
1068	. . . . .
1072	. . . . .
1076	. . . . .
1080	. . . . .
1084	. . . . .
1088	data: มีน, next: 1012
1092	. . . . .
1096	. . . . .
1100	. . . . .
1104	. . . . .
1108	. . . . .
1112	data: องศา, next: null
1116	. . . . .

Accessing Data

ถ้าคุณฉงนต้องการเข้าถึงข้อมูลตำแหน่งที่ 100,000

Linked List

```
loc = 100000;
i = 0;
while (i++<loc)
    node = node->next;
printf("%d", node->data);
```

- วิ่งไปเรื่อยๆ จนกว่าจะถึงตัวที่ 100,000
- ช้า

Array

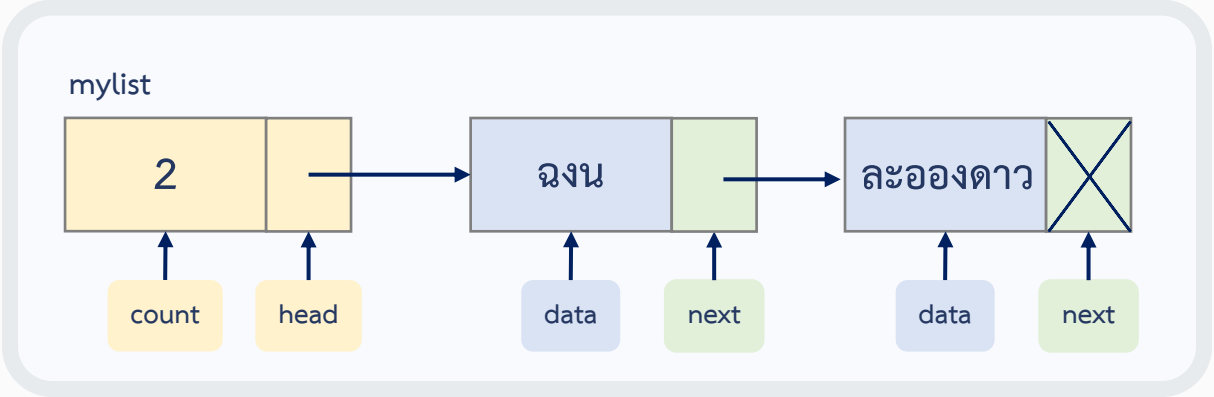
```
loc = 100000;
add = &num + (sizeof(int) * loc);
printf("%d", *add);
```

- คำนวณค่าของตำแหน่งข้อมูลได้เลย
- รู้ผลทันที

Linked List Lab

Lab Documentation

โครงสร้างของ Singly Linked List

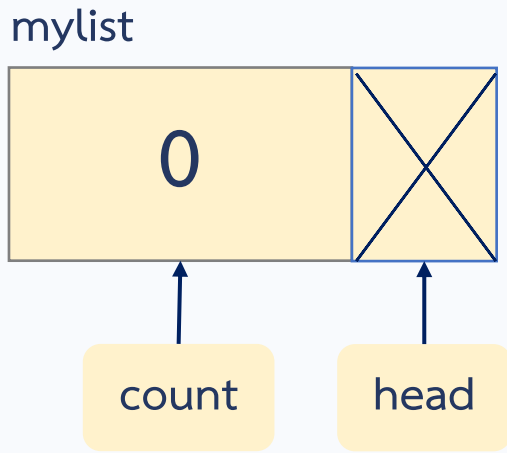



```
struct LinkedList {
  unsigned int count;
  struct DataNode* head;
};
```

```
struct DataNode {
  char* data;
  struct DataNode* next;
};
```

ตัวอย่างการสร้างอ็อบเจกต์ลิงค์ลิสต์และโหนด (1/5)

```
struct LinkedList {
  unsigned int count;
  struct DataNode* head;
};
```



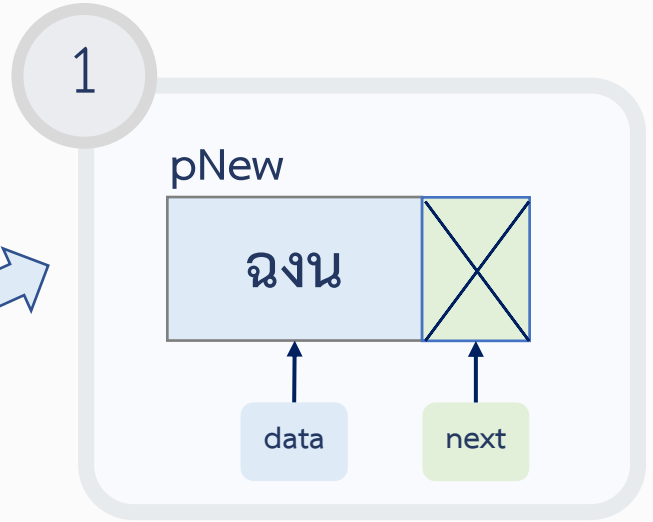

```
struct LinkedList* mylist = (struct LinkedList*)malloc(sizeof(struct LinkedList));
```

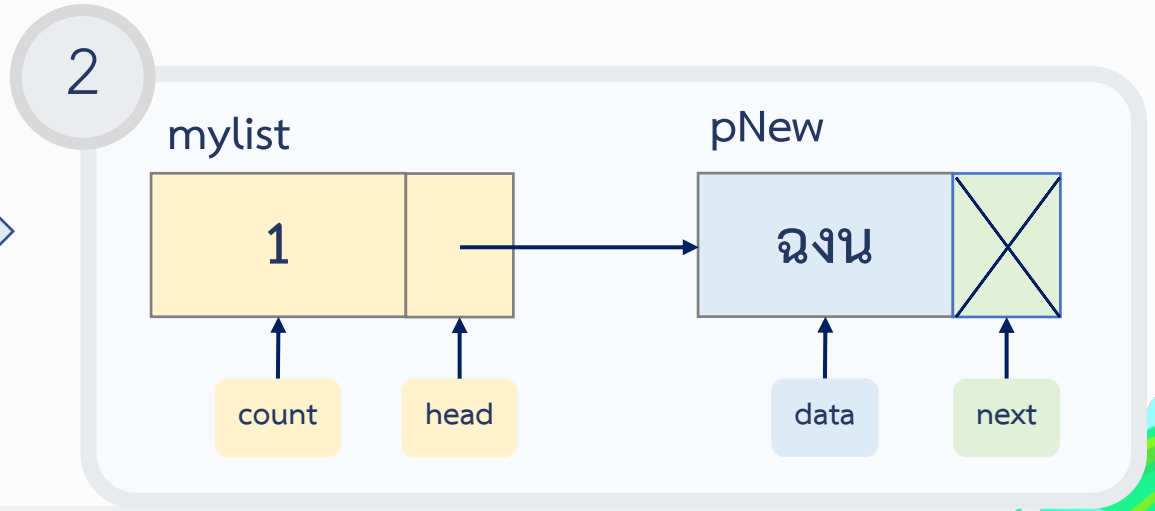
ตัวอย่างการสร้างอ็อบเจกต์ลิงค์ลิสต์และโหนด (2/5)

```
struct DataNode {
  char data;
  struct DataNode* next;
};
```

1 `struct DataNode* pNew = createDataNode("ฉงน");`

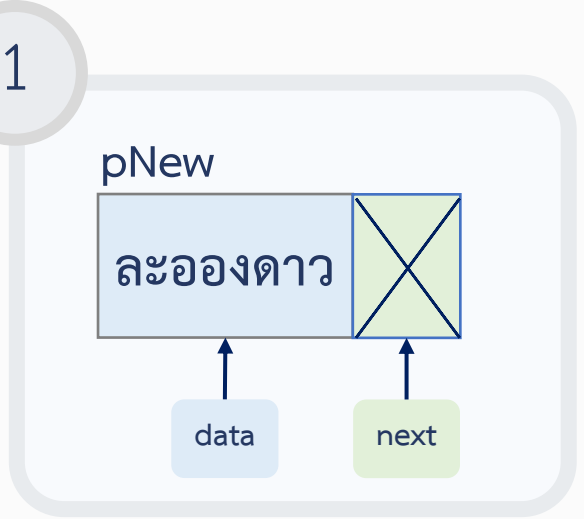
2
```
mylist->head = pNew;
mylist->count++;
```

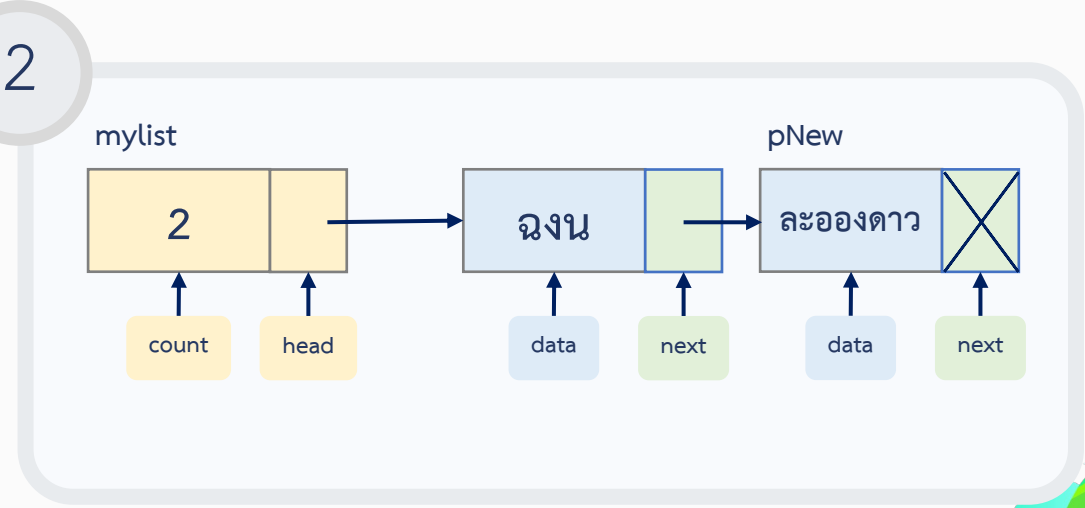




ตัวอย่างการสร้างอ็อบเจกต์ลิงค์ลิสต์และโหนด (3/5)

```
1  pNew = createDataNode("ละอองดาว");

2  mylist->head->next = pNew;
   mylist->count++;
```



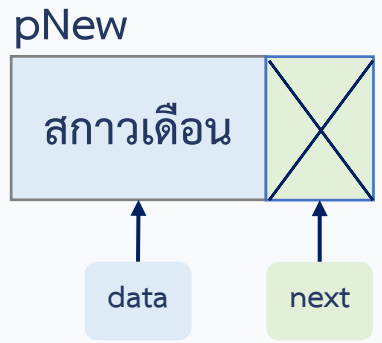


ตัวอย่างการสร้างอ็อบเจกต์ลิงค์ลิสต์และโหนด (4/5)

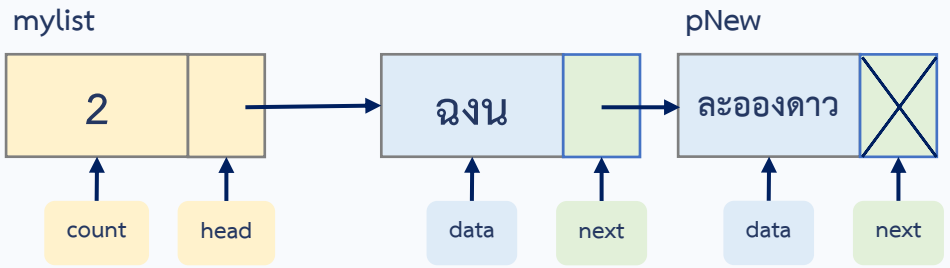
1
```
pNew = createDataNode("สกาวเดือน");
```

2
```
pNew->next = mylist->head->next;
mylist->head->next = pNew;
```

1



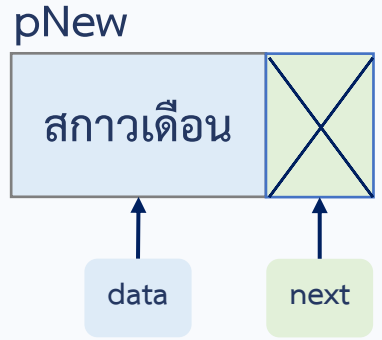
0



ตัวอย่างการสร้างอ็อบเจกต์ลิงค์ลิสต์และโหนด (4/5)

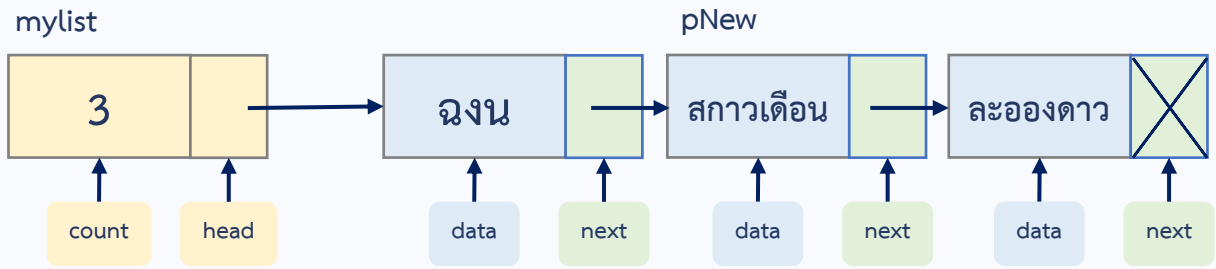
```
1  pNew = createDataNode("สกาวเดือน");

2  pNew->next = mylist->head->next;
   mylist->head->next = pNew;
```

1



ผลลัพธ์

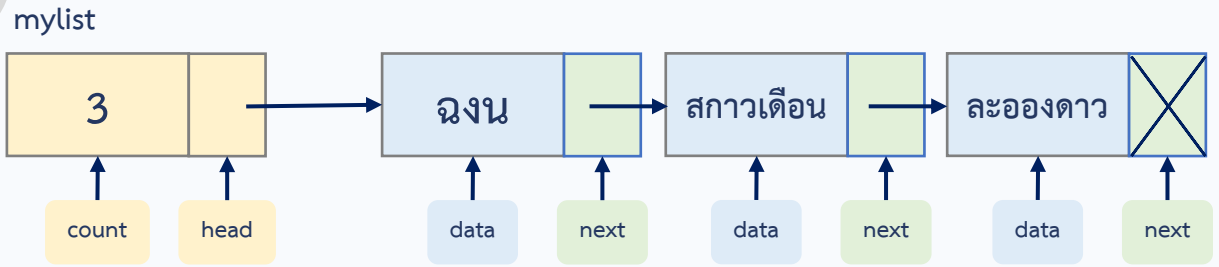
2



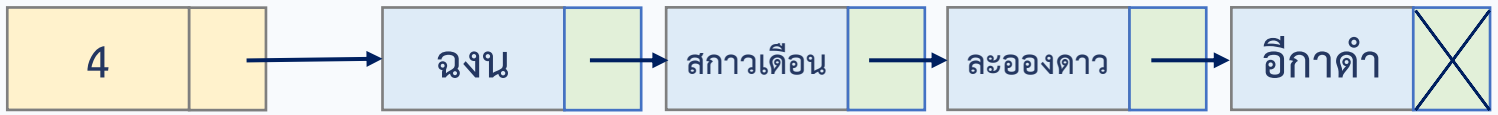
ตัวอย่างการสร้างอ็อบเจกต์ลิงค์ลิสต์และโหนด (5/5)

1
```
insert_last(mylist, "อีกาดำ");
```
2
```
delete(mylist, "สกาวเดือน");
```

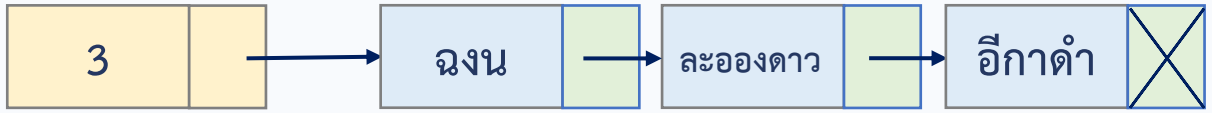
0



1



2

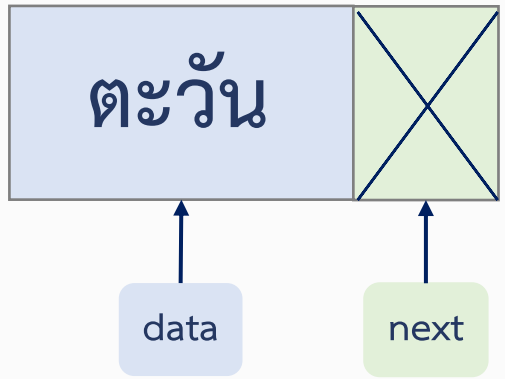


Labs 05.01 – Create DataNode

จงเขียนโปรแกรมเพื่อสร้างโครงสร้าง DataNode โครงสร้างนี้เป็นการจำลองโครงสร้างของ**โหนดข้อมูล**ที่ใช้เก็บข้อมูลและการเชื่อมต่อกับโหนดข้อมูลตัวถัดไปใน Singly Linked List ซึ่งเป็นโครงสร้างข้อมูลที่มีลักษณะเป็นลิงค์ของโหนดที่เชื่อมต่อกันในทิศทางเดียว โดยโหนดแต่ละตัวประกอบไปด้วยข้อมูลและหมายเลข Address ของโหนดถัดไป

มีคุณลักษณะ (Attributes) ดังนี้

- data: **ข้อมูลของโหนด** (เก็บข้อมูลเป็น String)
  - ต้องรับค่าข้อมูลผ่านฟังก์ชัน createDataNode(char* data)
- next: **หมายเลข Address ของโหนดถัดไป**
  - ค่าเริ่มต้นคือ None



เมื่อสร้าง DataNode เสร็จแล้วให้นำโค้ดจาก <e>Judge ไปรันทดสอบและส่งขึ้นไปตรวจ

ตัวอย่าง

- pNew = createDataNode(“ตะวัน”)
- print(pNew.data()) // ตะวัน
- print(pNew.next()) // None

Labs 05.02 – Singly Linked List (Traversal and Insert Last)

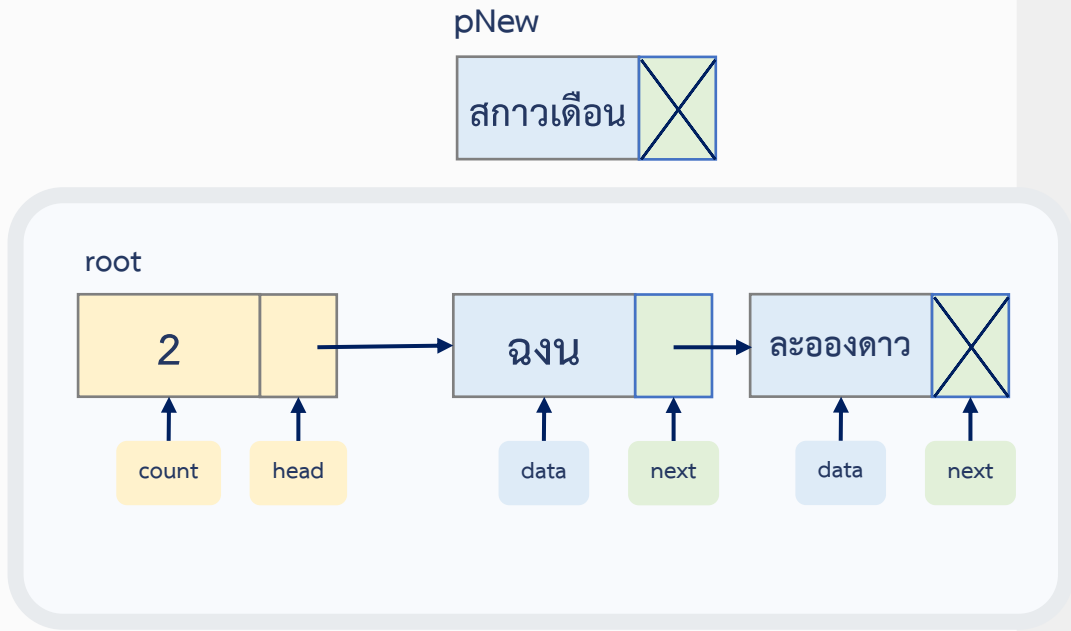
จงเขียนโปรแกรมเพื่อสร้างโครงสร้าง SinglyLinkedList ในการทำแลปครั้งนี้ ไปทีละขั้นตอน โดยเราจะเริ่มจากคุณลักษณะทั้งหมดและเมธอดหลักๆ 2 ตัวได้แก่

- traverse(mylist): **ท่องเข้าไปในลิงค์ลิสต์และแสดงข้อมูลตามลำดับทั้งหมดในลิงค์ลิสต์**
  - เชื่อมต่อกันด้วย -> เช่น ตะวัน -> ซันจา
  - กรณีที่ลิสต์ว่าง ให้แสดงข้อความ “This is an empty list.”
- insertLast(mylist, data): **สร้างโหนดข้อมูล (DataNode) และทำการแทรกโหนดนั้นไปที่ส่วนด้านท้ายของลิงค์ลิสต์**
  - เช่น insertLast(mylist, “นาดอส”) ; ผลลัพธ์จะออกมาเป็น traverse(mylist) = ตะวัน -> ซันจา -> นาดอส

Inserting Nodes at the End of a List

- ปัญหาคือ เรารู้ได้อย่างไรว่าโหนดไหนเป็นโหนดท้ายของลิสต์
  - ต้องท่อง (Traverse) เข้าไปในลิสต์จนถึงส่วนท้ายสุด

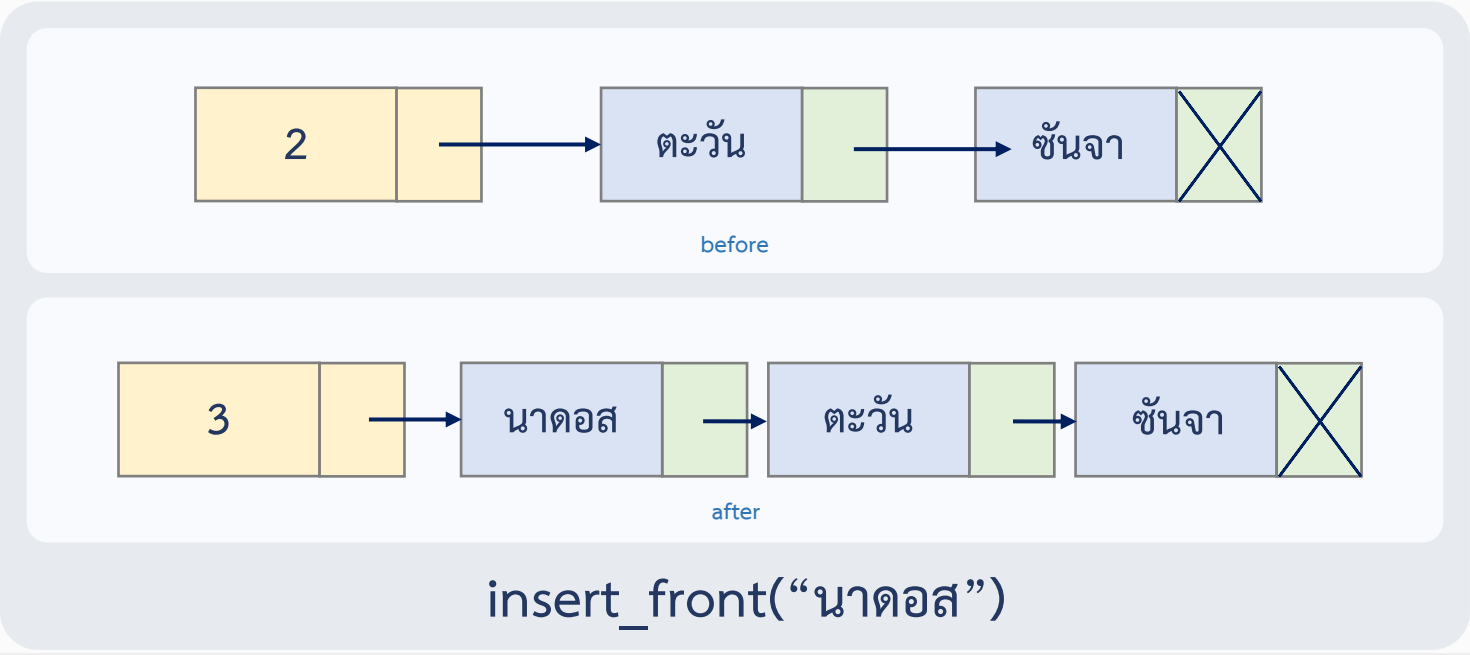
```
pNew = createDataNode(สกาวเดือน)
If (root.head is null)
        root.head = pNew
else
        start = root.head
        loop (start.next != null)
                start = start.next
        end loop
        start.next = pNew
end if
```



Labs 05.03 – Singly Linked List (Insert Front)

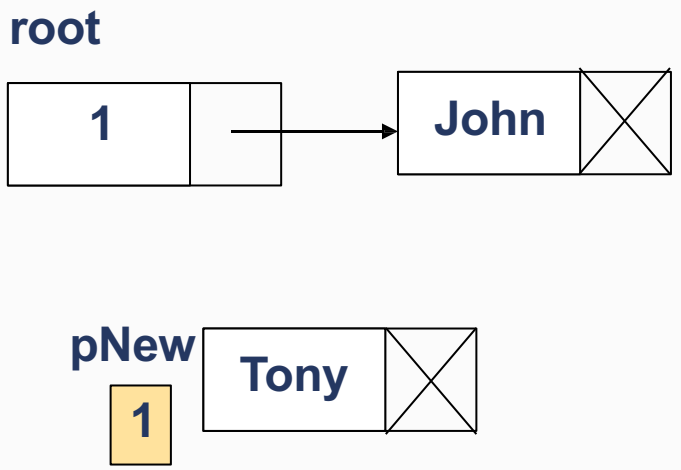
ในข้อนี้เราจะให้คุณเพิ่มฟังก์ชัน insert_front เข้าไปในคลาสที่คุณเขียนจากข้อก่อนหน้า

- insertFront(data): **สร้างโหนดข้อมูล (DataNode) และทำการแทรกโหนดนั้นไปที่ส่วนด้านหน้าของลิงค์ลิสต์**
  - เช่น insertFront(mylist, “นาดอส”) ; ผลลัพธ์จะออกมาเป็น traverse(mylist) = นาดอส -> ตะวัน -> ซันจา



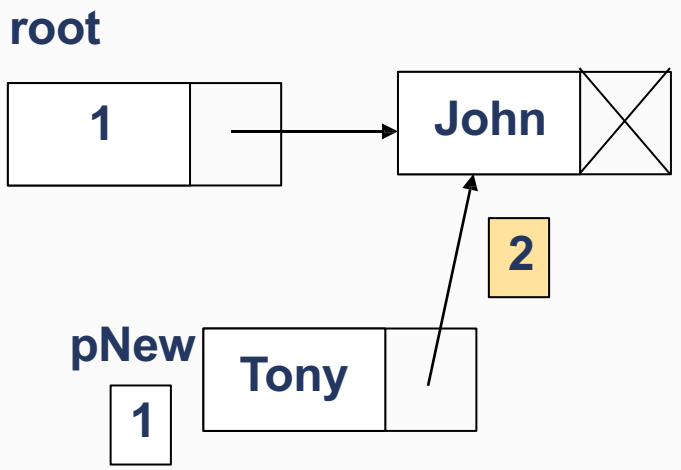
Inserting Nodes at the Front of a List

- ถ้าจะเพิ่มโหนดที่มีชื่อเป็น **Tony** ไว้ต้นลิสต์
  - *pNew = createDataNode(Tony,null)*
  - pNew.next = root.head
  - root.head = pNew
  - Increase root count



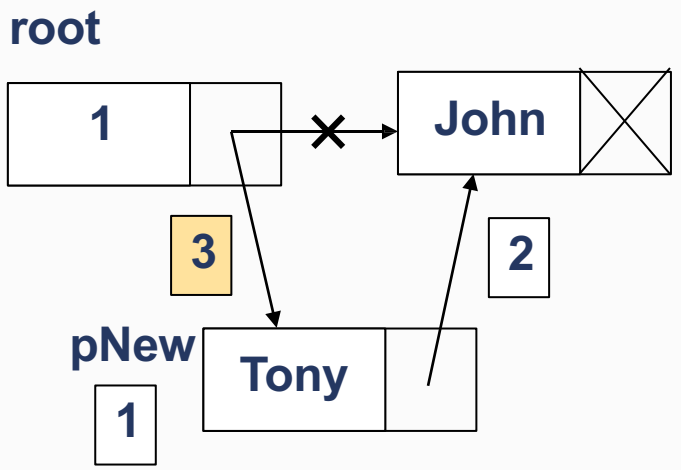
Inserting Nodes at the Front of a List

- ถ้าจะเพิ่มโหนดที่มีชื่อเป็น **Tony** ไว้ต้นลิสต์
  - pNew = createDataNode(Tony,null)
  - *pNew.next = root.head*
  - root.head = pNew
  - Increase root count



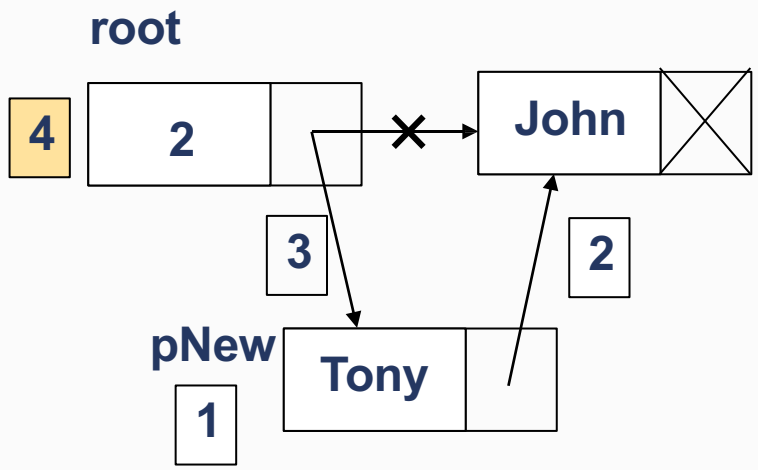
Inserting Nodes at the Front of a List

- ถ้าจะเพิ่มโหนดที่มีชื่อเป็น **Tony** ไว้ต้นลิสต์
  - pNew = createDataNode(Tony,null)
  - pNew.next = root.head
  - *root.head = pNew*
  - Increase root count



Inserting Nodes at the Front of a List

- ถ้าจะเพิ่มโหนดที่มีชื่อเป็น **Tony** ไว้ต้นลิสต์
  - pNew = createDataNode(Tony,null)
  - pNew.next = root.head
  - root.head = pNew
  - *Increase root count*



Labs 05.04 – Singly Linked List (Delete)

ในข้อนี้เราจะให้คุณเพิ่มฟังก์ชัน delete เข้าไปในคลาสที่คุณเขียนจากข้อก่อนหน้า

- delete(data): **ลบโหนดข้อมูล (DataNode) ออกจากลิงค์ลิสต์**
  - กรณีที่ไม่มีโหนดในลิสต์ ให้แสดงข้อความ “Cannot delete, <data> does not exist.”
  - เช่น delete(mylist, “ตะวัน”) ; ผลลัพธ์จะออกมาเป็น traverse(mylist) = ซันจา

delete(“ตะวัน”)

Deleting Nodes from an Unordered List

- กรณีที่ต้องการลบโหนดที่อยู่หัวลิสต์

root.head = root.head.next ; delete start;

- กรณีที่ต้องการลบโหนดที่อยู่ระหว่างลิสต์

  prev.next = start.next; delete start;

- กรณีที่ต้องการลบโหนดที่อยู่ท้ายลิสต์

  prev.next = NULL; delete start;