



รายงาน : Lab06 – Intro to Linked List

จัดทำโดย

นายกิตติพิศ หนูทอง รหัสนักศึกษา : 6135512003

นายปฏิภาณ วรรณโก รหัสนักศึกษา : 6135512059

Section : 01

240-207 Programing and Data Structures

“งานทั้งหมดนี้ในรายงานฉบับนี้ล้วนเป็นผลงานของข้าพเจ้า มิได้ลอกหรือสำเนาจากที่อื่นใดในกรณีที่พบว่าเกิดสำเนาด้วยวิธีใดก็ตาม ข้าพเจ้ายินดีไม่ขอรับคะแนนจากรายงานฉบับนี้”

คะแนนที่ได้

ลงชื่อ

กิตติพิศ หนูทอง
(นายกิตติพิศ หนูทอง)

ปฏิภาณ วรรณโก
(นายปฏิภาณ วรรณโก)

ข้อที่ 1 : Insert At Front

Insert At Front

รับข้อมูลตัวเลขจำนวนเต็ม จนกว่าผู้ใช้จะใส่ข้อมูลที่น้อยกว่าหรือเท่ากับ 0 นำข้อมูลใส่ linked list ในรูปแบบ "Insert at front" และทำการแสดงผลข้อมูล พร้อมหาค่าผลรวมของข้อมูลทั้งหมด

```
Enter: <1 3 2 7 4 -1>
= 4 7 2 3 1
= 17
```

Code

lab06-1.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct listnode
4  {
5      int data;
6      struct listnode *next;
7  };
8  typedef struct listnode LN;
9
10 void insert_at_front(LN **hptr, int d);
11 void print(LN *head);
12 int sum(LN *head);
13
14 int main()
15 {
16     LN *head=NULL;
17     int d;
18     printf("Enter: ");
19     do
20     {
21         scanf("%d",&d);
22         if(d > 0)
23         {
24             insert_at_front(&head,d);
25         }
26     }while(d > 0);
27     printf("= ");
28     print(head);
29
30     printf("\n= %d",sum(head));
31
32     return 0; }
33
```

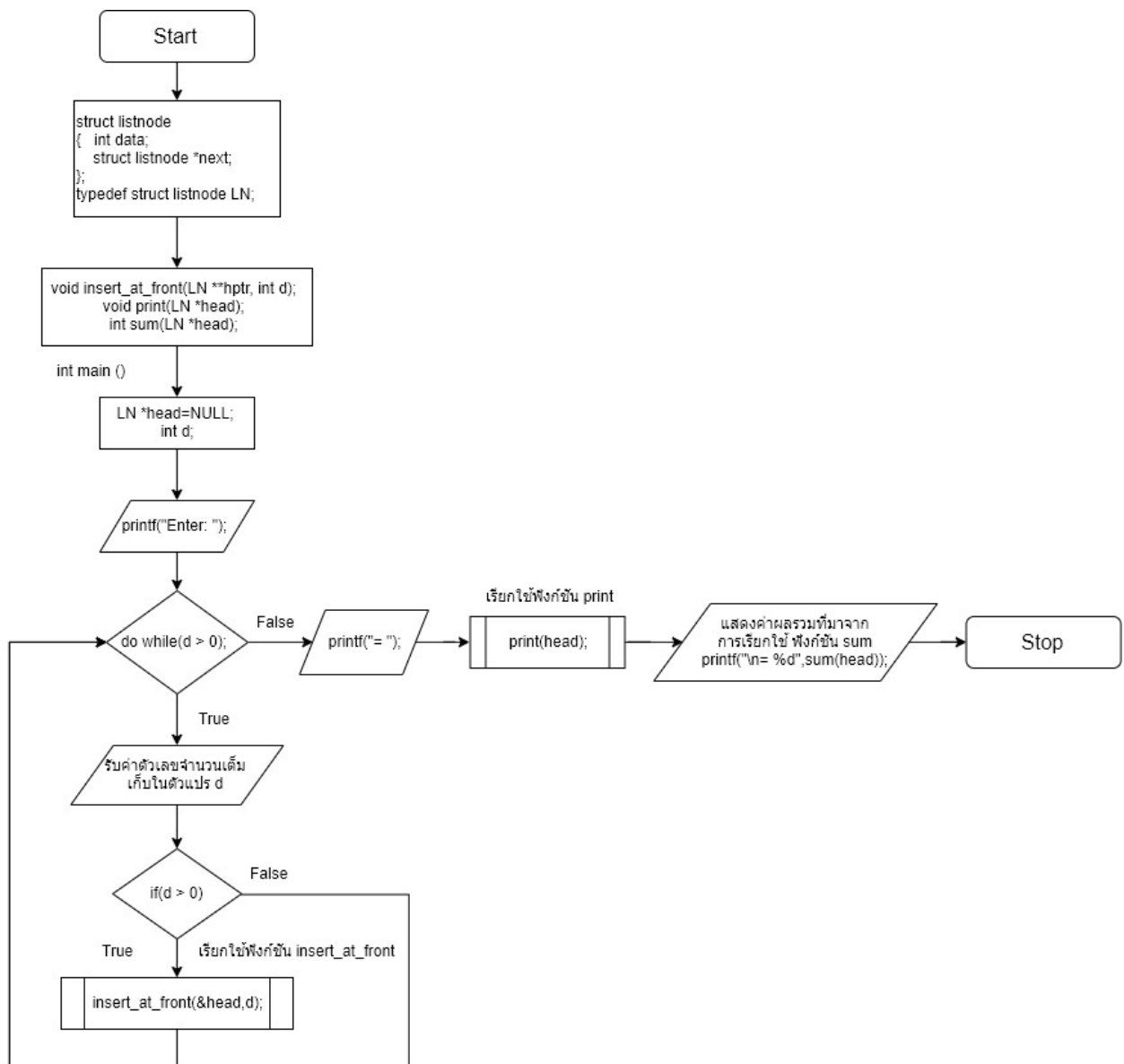
```

34 void insert_at_front(LN **hptr, int d)
35 {
36     LN *new_node = NULL;
37     new_node=(LN*)malloc(sizeof(LN));
38     new_node->data = d;
39     new_node->next = *hptr;
40     *hptr = new_node;
41 }
42 void print(LN *head)
43 {
44     while (head != NULL)
45     {
46         printf("%d ",head->data);
47         head = head->next;
48     }
49 }
50 int sum(LN *head)
51 {
52     int ssum=0;
53     while (head != NULL)
54     {
55         ssum += head->data;
56         head = head->next;
57     }
58     return ssum;
59 }

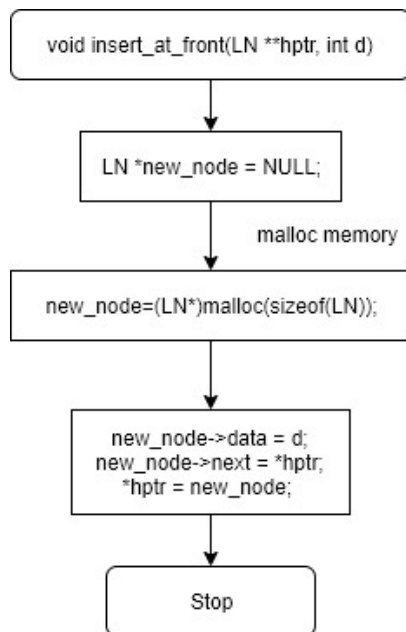
```

Flowchart

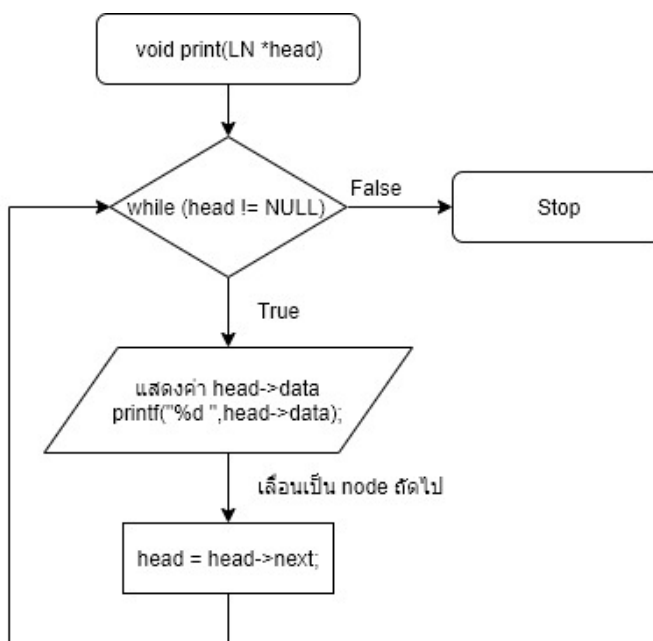
ฟังก์ชัน main()



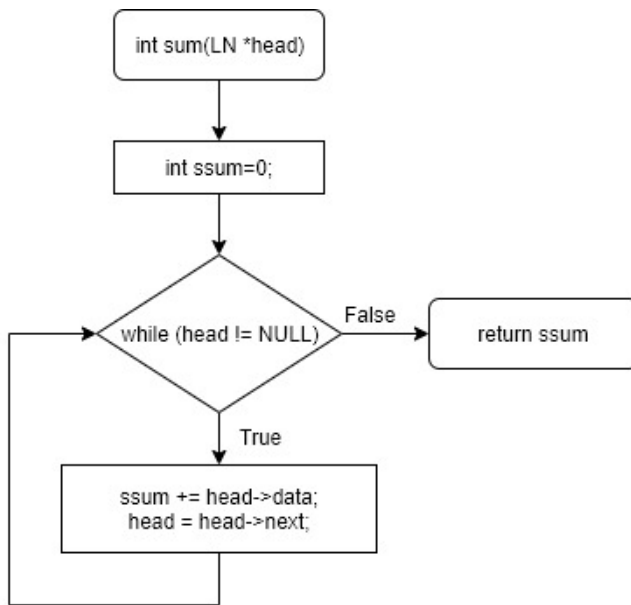
ฟังก์ชัน void insert_at_front(LN **hptr, int d)



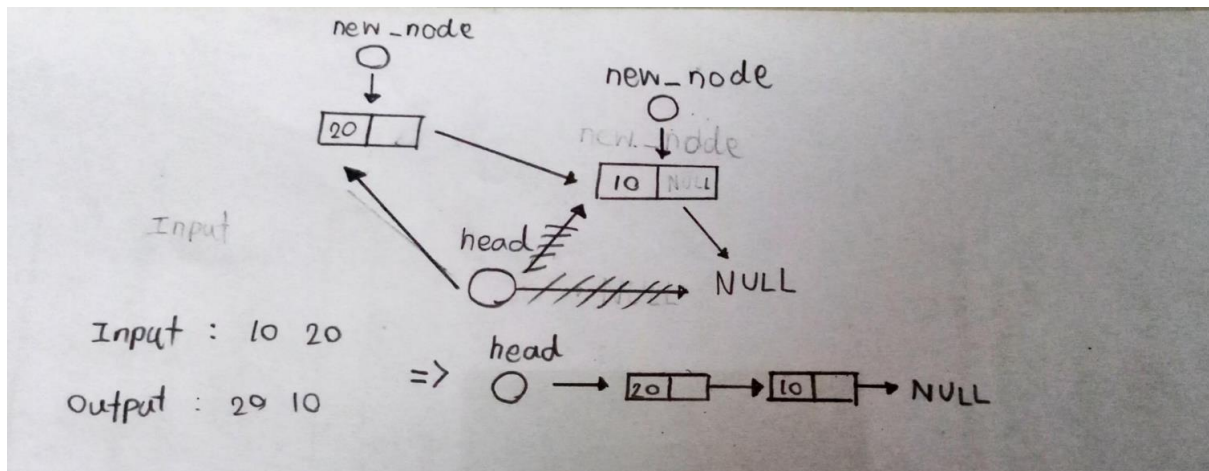
ฟังก์ชัน void print(LN *head)



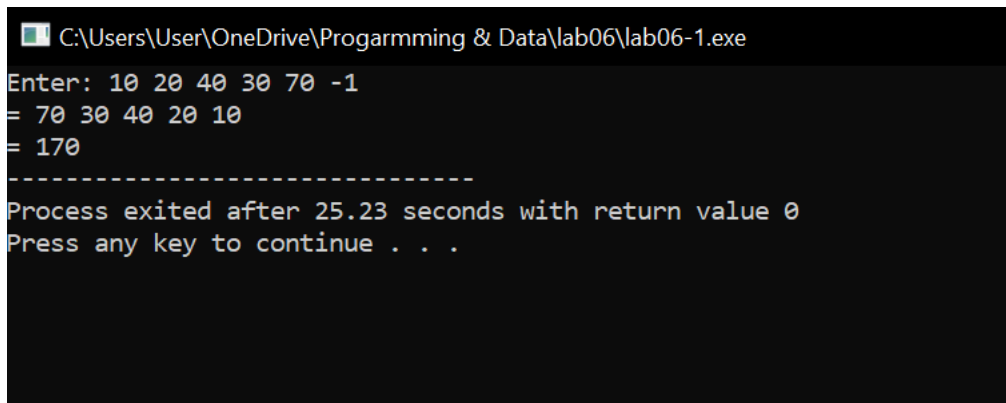
ฟังก์ชัน int sum(LN *head)



แผนภาพ Linked List



ผลการรันโปรแกรม



```
C:\Users\User\OneDrive\Programming & Data\lab06\lab06-1.exe
Enter: 10 20 40 30 70 -1
= 70 30 40 20 10
= 170
-----
Process exited after 25.23 seconds with return value 0
Press any key to continue . . .
```

อธิบายหลักการทำงาน : เป็นโปรแกรมที่รับค่าของจำนวนเต็ม จนกว่าผู้ใช้จะใส่ข้อมูลที่น้อยกว่าหรือเท่ากับ 0 โดยนำข้อมูลใส่ linked list ในรูปแบบ Insert at front แล้วทำการแสดงผลข้อมูลและหาค่าผลรวมของตัวเลขทั้งหมดออกมาทางหน้าจอ ในโปรแกรมจะมีการประกาศโครงสร้างข้อมูล structure listnode ที่มีตัวแปร int data และ struct listnode *next ไว้สำหรับชี้ไปที่ node ถัดไป จากนั้น typedef struct listnode ให้เป็นชนิดตัวแปร LN

ภายในโปรแกรมจะประกอบไปด้วยฟังก์ชันหลัก main() และฟังก์ชันย่อยอีกสามฟังก์ชันโดยที่แต่ละฟังก์ชันจะทำงานไม่เหมือนกัน ดังนี้

ฟังก์ชัน main () จะทำการรับค่าจากผู้ใช้และส่งค่าไปยังฟังก์ชันอื่น ๆ เพื่อทำงานตามคำสั่งที่ได้เขียนไว้ในแต่ละฟังก์ชัน โดยฟังก์ชัน main () จะมีการประกาศตัวแปร pointer LN *head = NULL ไว้เป็น HEAD และตัวแปร int d ไว้เก็บค่าตัวเลขที่รับจากผู้ใช้ และทำการวนลูป do while (d > 0) ภายในวนลูปจะทำการรับค่าตัวเลขจนกว่าผู้ใช้จะป้อนค่าน้อยกว่าหรือเท่ากับ 0 ซึ่งในลูปจะมีการเรียกใช้ฟังก์ชันเพิ่มข้อมูล link list insert_at_front(&head,d); โดยจะส่งค่าของ address ของ pointer head และค่าของ d ไป เมื่อออกจากลูป จะมีการเรียกใช้ฟังก์ชันแสดงค่า print(head); โดยส่งค่าของ head ไปจากนั้นทำการแสดงค่าผลรวมที่มาจาก การเรียกใช้ฟังก์ชันผลรวม(sum) โดยส่งค่าของ head ไป ออกมาทางหน้าจอ

ฟังก์ชัน void insert_at_front(LN **hptr, int d) เป็นฟังก์ชันที่ทำหน้าที่ในการเพิ่มข้อมูลลงใน node ถัดไปของ linked list โดยที่จะมีการเพิ่มตัวแปร pointer LN * new_node และทำการ malloc ขอ memory ให้กับ pointer new_node ในขณะรันโปรแกรม จากนั้นให้ค่าของ new_node->date = d และ new_node->next = *hptr และ *hptr = new_node ซึ่งเป็นการเพิ่มข้อมูลไปทางด้านหน้าของ node

ฟังก์ชัน void print(LN *head) เป็นฟังก์ชันที่ทำหน้าที่ในการแสดงค่าจำนวนเต็มของ data ที่บันทึกไว้ในแต่ละ node ออกมาใช้ โดยการใช้ประโยชน์จากการเชื่อมกัน ซึ่ง node ที่เก็บค่าตัวเลขที่รับมาท้ายสุดจะแสดงค่าออกมาก่อน ในการแสดงค่าจะใช้การวนลูป while เงื่อนไข คือ head != NULL ในลูปจะแสดงค่าของ data ในแต่ละ node ออกมาและให้ head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวนลูปเลื่อน node ไปเรื่อย ๆ จนกว่าเจอ NULL

ฟังก์ชัน int sum(LN *head) จะมีประกาศตัวแปร int ssum = 0 จากนั้นวนลูปเงื่อนไข while (head != NULL) ในลูปจะใช้คำสั่ง ssum += head->data; เพื่อหาผลรวมของ data ในแต่ละ node และเก็บค่าไว้ในตัวแปร ssum ต่อมาใช้คำสั่ง head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวนลูปจนกว่าจะ NULL หลังจากนั้นคืนค่า ssum ให้กับฟังก์ชัน main ()

ความรู้จากการทำ Lab Insert At Front : ได้ความรู้ในการใส่ข้อมูลลงใน linked list ที่สามารถเพิ่มข้อมูลไปทางด้านหน้าได้ รวมถึงการแสดงผลและการเลื่อน node ของ linked list ซึ่งจะมี head เป็น node แรก โดยจะมีการใช้คำสั่ง head = head->next เพื่อเป็นการเลื่อนไปยัง node ถัดไป

ข้อที่ 2 : Insert At Back

Insert At Back

รับข้อมูลตัวเลขจำนวนเต็ม จนกว่าผู้ใช้จะใส่ข้อมูลที่น้อยกว่าหรือเท่ากับ 0 นำข้อมูลใส่ linked list ในรูปแบบ "Insert at back" และทำการแสดงผลข้อมูล พร้อมหาค่าผลรวมของข้อมูลทั้งหมด

```
Enter: <1 3 2 7 4 -1>
= 1 3 2 7 4
= 17
```

สามารถนิยามฟังก์ชันเพิ่มเติมได้

Code

lab06-2.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct listnode
4  {
5      int data;
6      struct listnode *next;
7  };
8  typedef struct listnode LN;
9
10 void insert_at_back(LN **hptr, int d);
11 void print(LN *head);
12 int sum(LN *head);
13
14 int main()
15 {
16     LN *head=NULL;
17     int d;
18     printf("Enter: ");
19     do
20     {
21         scanf("%d",&d);
22         if(d > 0)
23         {
24             insert_at_back(&head,d);
25         }
26     }while(d > 0);
27     printf("= ");
28     print(head);
29
30     printf("\n= %d",sum(head));
31
32     return 0; }
33
```

```

34 void insert_at_back(LN **hptr, int d)
35 {
36     LN *new_node = NULL;
37     LN *node_p = *hptr;
38     new_node=(LN*)malloc(sizeof(LN));
39     new_node->data = d;
40     new_node->next = NULL;
41     if(*hptr==NULL)
42     {
43         *hptr=new_node;
44     }
45     else
46     {
47         while(node_p->next!= NULL)
48         {
49             node_p = node_p->next;
50         }
51         node_p->next = new_node;
52     }
53 }
54

```

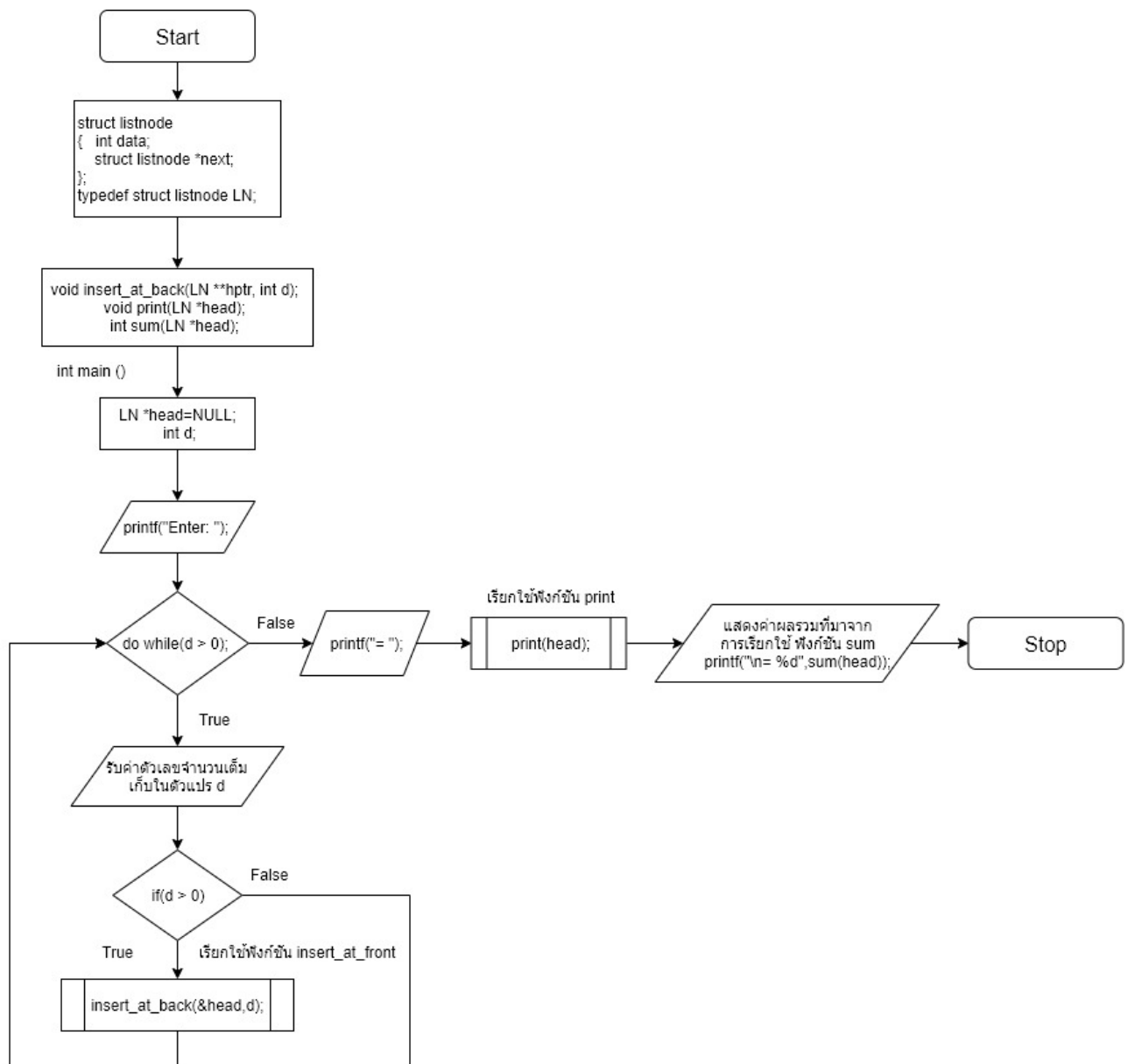
```

55 void print(LN *head)
56 {
57     while (head != NULL)
58     {
59         printf("%d ",head->data);
60         head = head->next;
61     }
62 }
63 int sum(LN *head)
64 {
65     int ssum=0;
66     while (head != NULL)
67     {
68         ssum += head->data;
69         head = head->next;
70     }
71     return ssum;
72 }

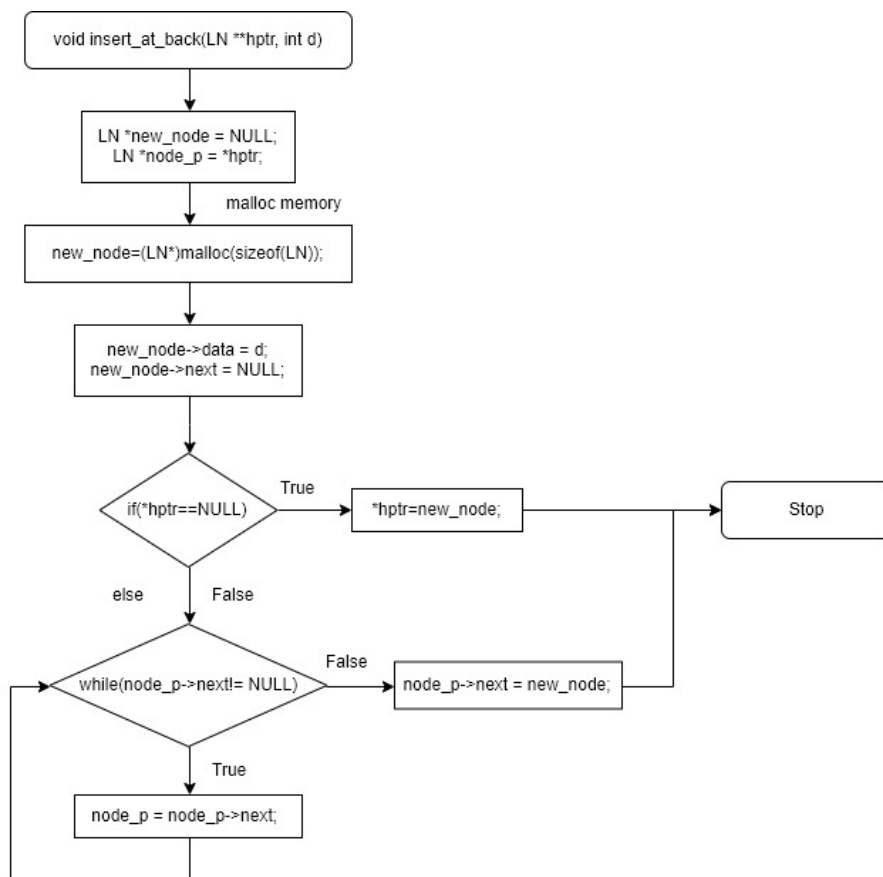
```

Flowchart

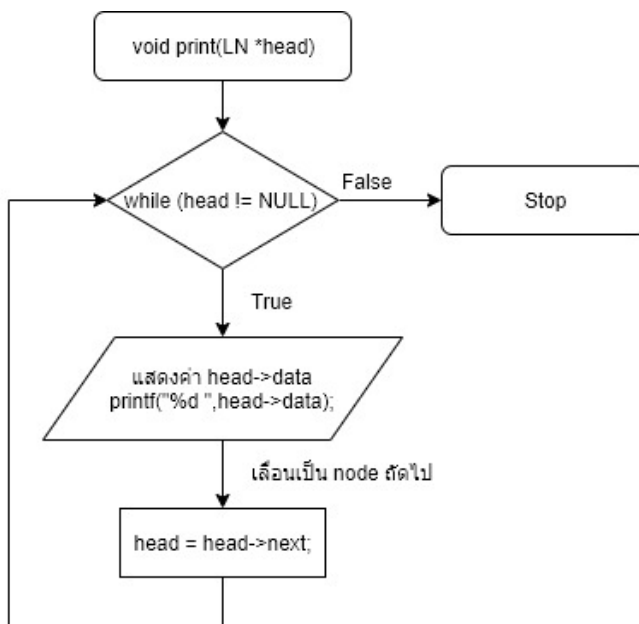
ฟังก์ชัน main()



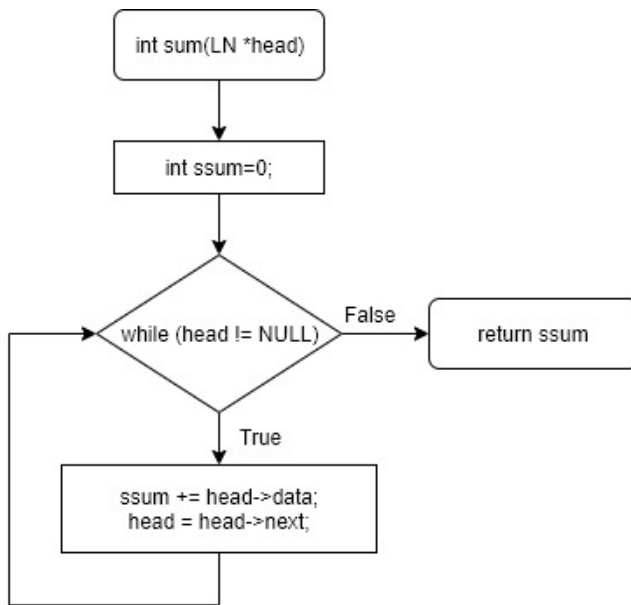
ฟังก์ชัน void insert_at_back(LN **hptr, int d)



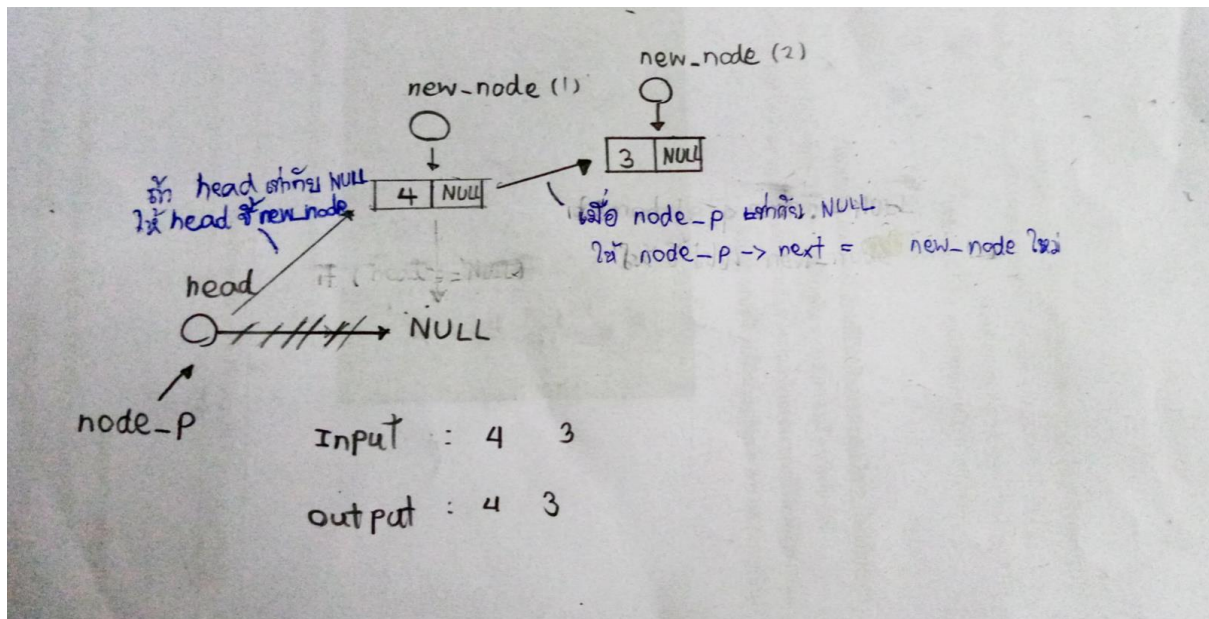
ฟังก์ชัน void print(LN *head)



ฟังก์ชัน int sum(LN *head)



แผนภาพ Linked List



ผลการรันโปรแกรม

```
C:\Users\User\OneDrive\Programing & Data\lab06\lab06-2.exe
Enter: 4 3 7 11 2 -1
= 4 3 7 11 2
= 27
-----
Process exited after 16.78 seconds with return value 0
Press any key to continue . . .
```

อธิบายหลักการทำงาน : เป็นโปรแกรมที่รับค่าของจำนวนเต็ม จนกว่าผู้ใช้จะใส่ข้อมูลที่น้อยกว่าหรือเท่ากับ 0 โดยนำข้อมูลใส่ link list ในรูปแบบ Insert at back แล้วทำการแสดงผลข้อมูลและหาค่าผลรวมของตัวเลขทั้งหมดออกมาทางหน้าจอ ในโปรแกรมจะมีการประกาศโครงสร้างข้อมูล structure listnode ที่มีตัวแปร int date และ struct listnode *next ไว้สำหรับชี้ไปที่ node ถัดไป จากนั้น typedef struct listnode ให้เป็นชนิดตัวแปร LN

ภายในโปรแกรมจะประกอบไปด้วยฟังก์ชันหลัก main() และฟังก์ชันย่อยอีกสามฟังก์ชันโดยที่แต่ละฟังก์ชันจะทำงานไม่เหมือนกัน ดังนี้

ฟังก์ชัน main () จะทำการรับค่าจากผู้ใช้และส่งค่าไปยังฟังก์ชันอื่น ๆ เพื่อทำงานตามคำสั่งที่ได้เขียนไว้ในแต่ละฟังก์ชัน โดยฟังก์ชัน main () จะมีการประกาศตัวแปร pointer LN *head = NULL ไว้เป็น HEAD และตัวแปร int d ไว้เก็บค่าตัวเลขที่รับจากผู้ใช้ และทำการวนลูป do while (d > 0) ภายในวนลูปจะทำการรับค่าตัวเลขจนกว่าผู้ใช้จะป้อนค่าน้อยกว่าหรือเท่ากับ 0 ซึ่งในลูปจะมีการเรียกใช้ฟังก์ชันเพิ่มข้อมูล link list insert_at_back(&head,d); โดยจะส่งค่าของ address ของ pointer head และค่าของ d ไป เมื่อออกจากลูป จะมีการเรียกใช้ฟังก์ชันแสดงค่า print(head); โดยส่งค่าของ head ไปจากนั้นทำการแสดงค่าผลรวมที่มาจากการเรียกใช้ฟังก์ชันผลรวม(sum) โดยส่งค่าของ head ไป ออกมาทางหน้าจอ

ฟังก์ชัน void insert_at_back(LN **hptr, int d) เป็นฟังก์ชันที่ทำหน้าที่ในการเพิ่มข้อมูลลงใน node ถัดไปของ linklist โดยจะมีการประกาศตัวแปร pointer LN * new_node กับ pointer LN*node_p ให้ node_p = *hptr เพื่อให้ node_p ชี้ไปที่ให้ *hptr และทำการ malloc ขอ memory ให้กับ pointer new_node ในขณะรันโปรแกรม จากนั้นให้ค่าของ new_node->date = d และ new_node->next = NULL ใช้เงื่อนไข if (*hptr==NULL) เพื่อที่จะตรวจสอบว่าค่า *hptr เป็น node แรกหรือไม่ ถ้าใช่ให้ *hptr == new_node เมื่อไม่ตรงเงื่อนไขให้เริ่มจากการวนลูปเงื่อนไข while (node_p->next != NULL)

โดยในที่นี้คือการเช็ค node ตัวล่าสุดที่เพิ่มเข้ามาจากการเรียกฟังก์ชันครั้งก่อน แล้วในลูปจะให้ค่าของ node_p = node_p->next เพื่อให้ค่า node_p เลื่อนไปเรื่อย ๆ จนกว่าจะเป็นค่าตัวท้ายสุดหรือมีค่าเท่ากับ NULL เมื่อวนลูปครบเงื่อนไขให้ค่าของ node_p->next = new_node เพื่อเป็นการเพิ่มข้อมูลไปทางด้านหลังหรือ insert at back

ฟังก์ชัน void print(LN *head) เป็นฟังก์ชันที่ทำหน้าที่ในการแสดงค่าจำนวนเต็มของ data ที่บันทึกไว้ในแต่ละ node ออกมาใช้ โดยการใช้ประโยชน์จากการเชื่อมกัน ซึ่ง node ที่เก็บค่าตัวเลขที่รับมาแรกสุดจะแสดงค่าออกมาก่อน ในการแสดงค่าจะใช้การวนลูป while เงื่อนไข คือ head != NULL ในลูปจะแสดงค่าของ data ในแต่ละ node ออกมาและให้ head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวนลูปเลื่อน node ไปเรื่อย ๆ จนกว่าเจอ NULL

ฟังก์ชัน int sum(LN *head) จะมีประกาศตัวแปร int ssum = 0 จากนั้นวนลูปเงื่อนไข while (head != NULL) ในลูปจะใช้คำสั่ง ssum += head->data; เพื่อหาผลรวมของ data ในแต่ละ node และเก็บค่าไว้ในตัวแปร ssum ต่อมาใช้คำสั่ง head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวนลูปจนกว่าจะ NULL หลังจากนั้นคืนค่า ssum ให้กับฟังก์ชัน main ()

ความรู้จากการทำ Lab Insert At Back : ได้ความรู้ในเรื่องการเพิ่มข้อมูลลงไปใน Linked List ในรูปแบบ Insert At Back ซึ่งจะเป็นการเลื่อนหา node ไปเรื่อย ๆ จนเจอ node ตัวสุดท้ายจากนั้นต่อท้าย node ที่เก็บข้อมูลท้ายสุด