



รายงาน : Lab07 – More Linked List

จัดทำโดย

นายกิตติพิศ หนูทอง รหัสนักศึกษา : 6135512003

นายปฏิภาณ วรรณโก รหัสนักศึกษา : 6135512059

Section : 01

240-207 Programing and Data Structures

“งานทั้งหมดนี้ในรายงานฉบับนี้ล้วนเป็นผลงานของข้าพเจ้า มิได้ลอกหรือสำเนาจากที่อื่นใดในกรณีที่พบว่าเกิดสำเนาด้วยวิธีใดก็ตาม ข้าพเจ้ายินดีไม่ขอรับคะแนนจากรายงานฉบับนี้”

คะแนนที่ได้

ลงชื่อ

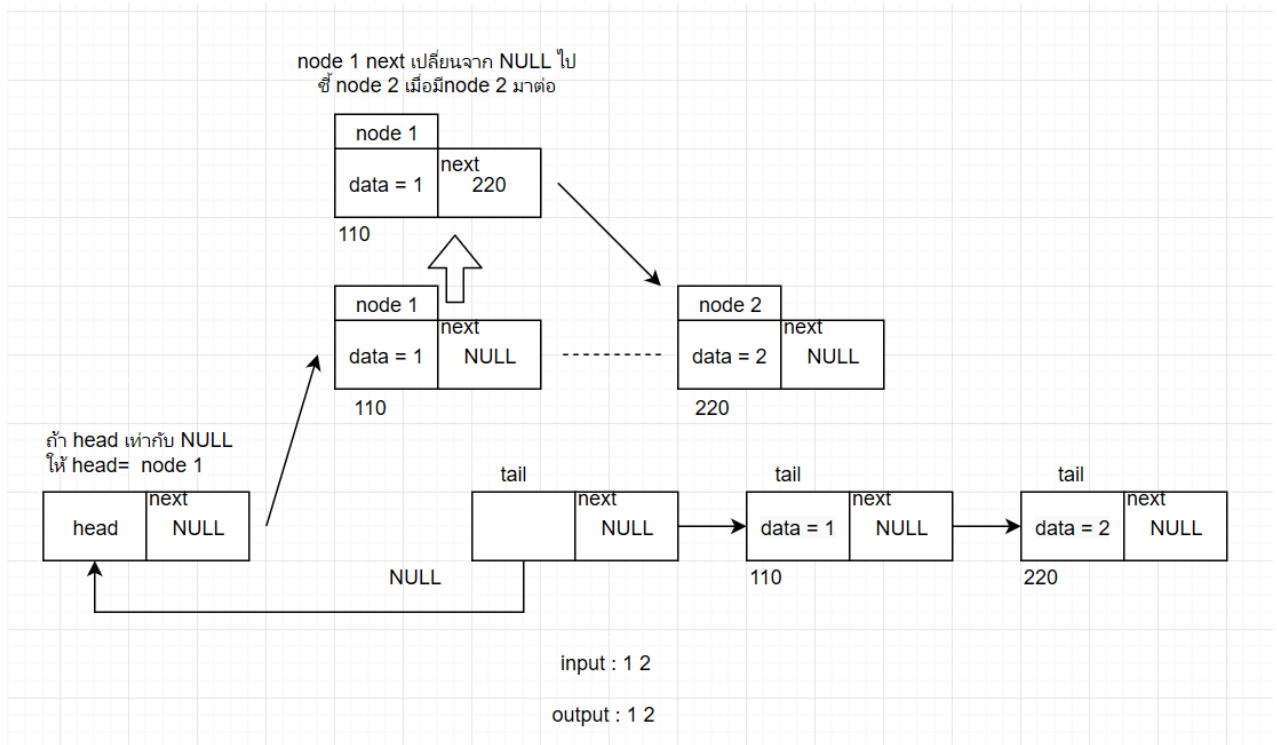
กิตติพิศ หนูทอง
(นายกิตติพิศ หนูทอง)

ปฏิภาณ วรรณโก
(นายปฏิภาณ วรรณโก)

Lab07 – More Linked List

แผนภาพ Linked List แบบ Insert At Back

แนวคิด : เป็นการค้นหา node ท้ายของ Linked List เพื่อนำ node ที่เก็บค่าข้อมูลที่เพิ่มมาใหม่ ไปต่อท้ายเป็น node ถัดไป



ข้อที่ 1 : Linked Song 1

Linked Song 1

รับข้อมูล Song จากผู้ใช้จำนวน 5 เพลง เก็บไว้ในลิงค์ลิสต์ โดยใช้เทคนิค insert at back โดยให้ถือว่าชื่อเพลงไม่มีเว้นวรรค
เขียนโค้ดโดยใช้ต้นแบบของฟังก์ชันที่เหมาะสม

```
S01: <I_Love_You 180.4>
...
= 3120.50
```

Code

lab07-1.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct song
4  {
5      char title[128];
6      double duration;
7  };
8  typedef struct song Song;
9
10 struct listnode
11 {
12     Song s;
13     struct listnode *next;
14 };
15 typedef struct listnode LN;
16
17 void insert(LN **hptr, Song *sp);
18 LN *find_tail(LN *head);
19 void print(LN *head);
20 double sum_duration(LN *head);
21
22 int main()
23 {
24     LN *head=NULL;
25     Song buf;
26     int i;
27     for(i=0;i<5;i++)
28     {
29         fflush(stdin);
30         printf("Song[%0d] : ",i+1);
31         scanf("%[^\n]s",buf.title);
32         printf("Duration : ");
33         scanf("%lf",&buf.duration);
34         insert(&head,&buf);
35     }
36     printf("\n");
37     print(head);
38     printf("\nTotal duration = %.2lf",sum_duration(head));
39     return 0;
40 }
41
```

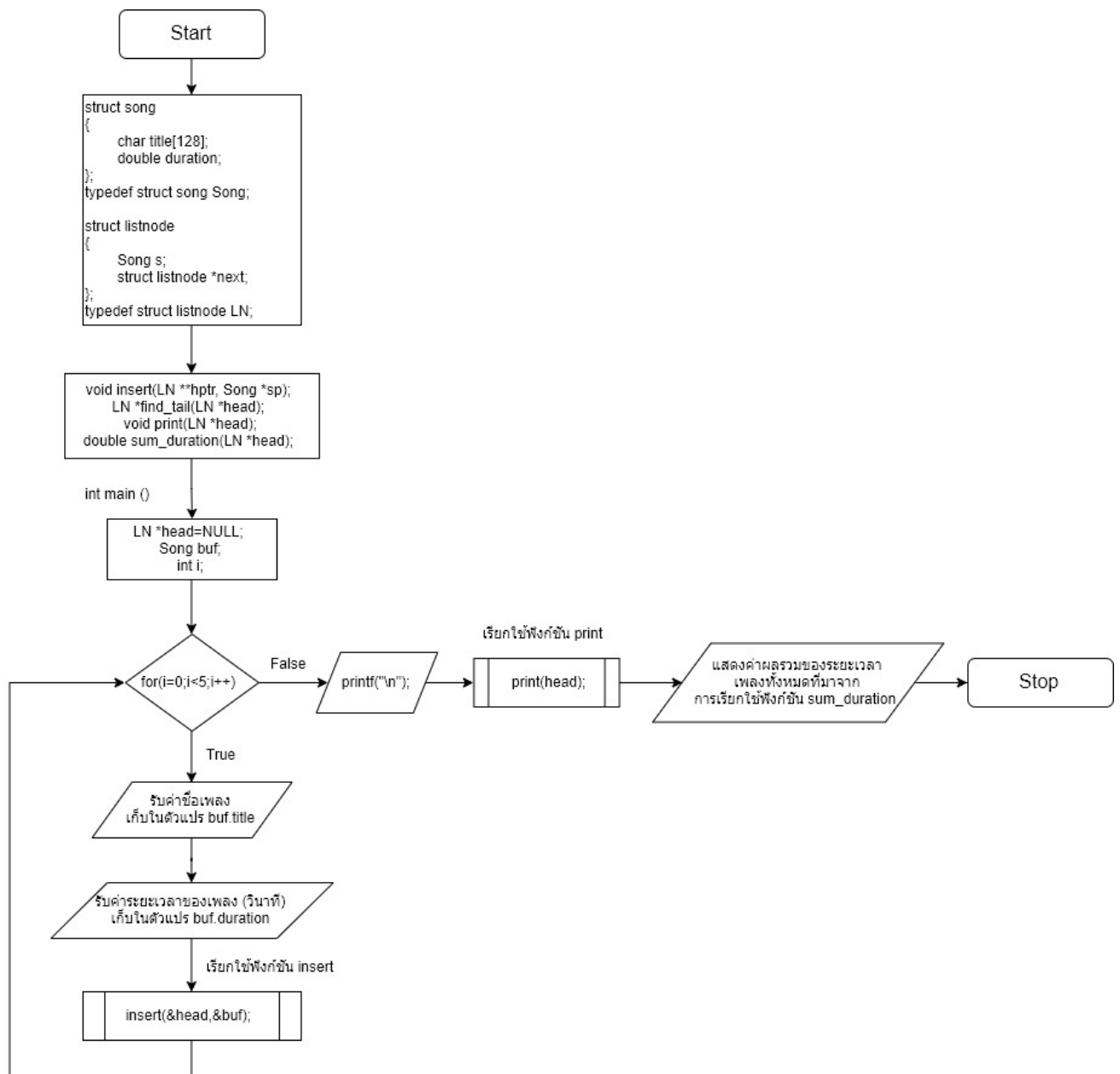
```

42 void insert(LN **hptr, Song *sp)
43 {
44     LN *new_node;
45     LN *tail;
46     new_node=(LN*)malloc(sizeof(LN));
47     new_node->s = *sp;
48     new_node->next = NULL;
49
50     tail = find_tail(*hptr);
51     if(*hptr==NULL)
52     {
53         *hptr=new_node;
54     }
55     else
56     {
57         tail->next = new_node;
58     }
59 }
60
61 LN *find_tail(LN *head)
62 {
63     LN *tails;
64     if(head == NULL)
65     {
66         return NULL;
67     }
68     else
69     {
70         tails = head;
71         while(tails->next!= NULL)
72         {
73             tails = tails->next;
74         }
75         return tails;
76     }
77 }
--
79 void print(LN *head)
80 {
81     int i=1;
82     while (head != NULL)
83     {
84         printf("%d:%s ",i,head->s.title);
85         head = head->next;
86         i++;
87     }
88 }
89
90 double sum_duration(LN *head)
91 {
92     double ssum=0;
93     while (head != NULL)
94     {
95         ssum += head->s.duration;
96         head = head->next;
97     }
98     return ssum;
99 }

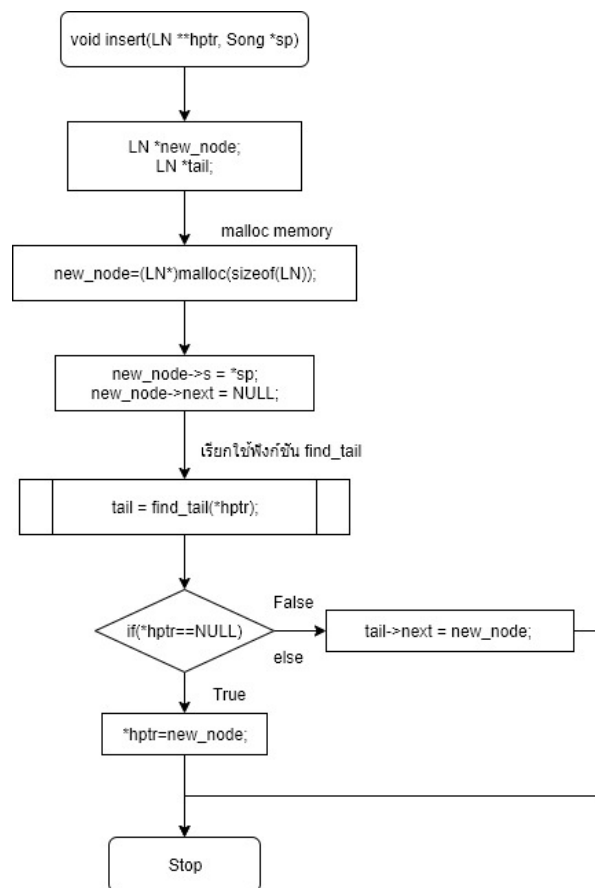
```

Flowchart

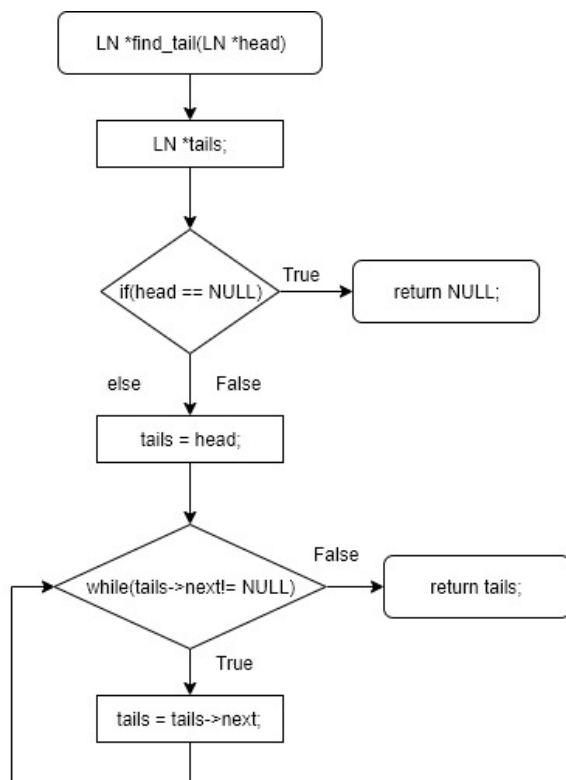
ฟังก์ชัน main()



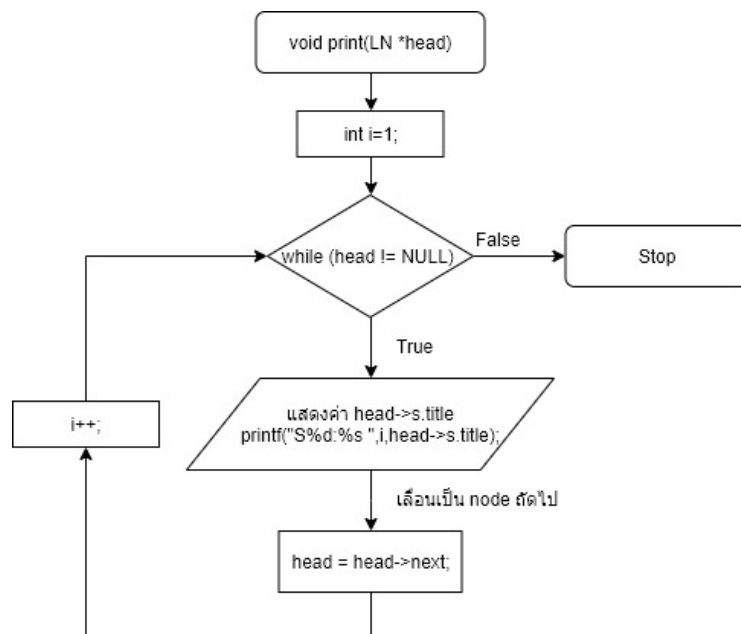
ฟังก์ชัน void insert(LN **hptr, Song *sp)



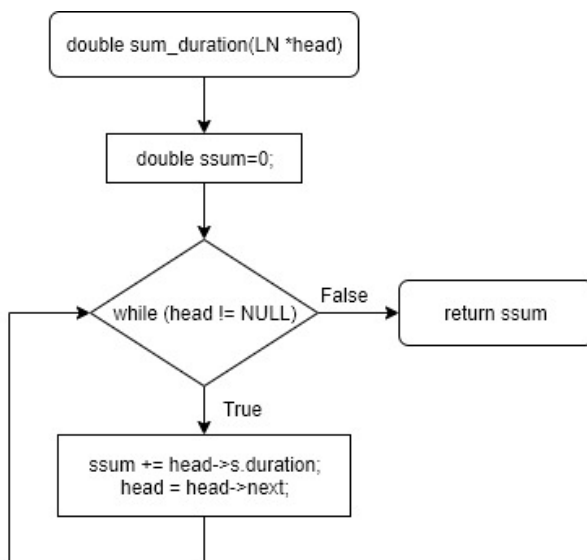
ฟังก์ชัน LN *find_tail(LN *head)



ฟังก์ชัน void print(LN *head)



ฟังก์ชัน double sum_duration(LN *head)



ผลการรันโปรแกรม

```
C:\Users\User\OneDrive\Programming & Data\lab07\lab07-1.exe
Song[1] : Shine_Your_Way
Duration : 216
Song[2] : Something_Just_Like_This
Duration : 396
Song[3] : Closer
Duration : 243
Song[4] : How_Do_You_Sleep?
Duration : 228
Song[5] : Thunder
Duration : 204

S1:Shine_Your_Way S2:Something_Just_Like_This S3:Closer S4:How_Do_You_Sleep? S5:Thunder
Total duration = 1287.00
-----
Process exited after 70.05 seconds with return value 0
Press any key to continue . . .
```

อธิบายหลักการทำงาน : เป็นโปรแกรมที่รับค่าข้อมูลเพลง 5 จำนวนที่ประกอบไปด้วยชื่อและจำนวนเวลาหน่วยเป็นวินาที โดยใช้ link list แบบ insert at back โดยสร้างโครงสร้างข้อมูล Struct song ภายในโครงสร้างจะมีตัวแปรเก็บค่าชื่อเพลง char title[128] และตัวแปรเก็บค่าจำนวนเวลา double duration; จากนั้น typedef struct song Song; ต่อมาสร้างโครงสร้างข้อมูล struct listnode ที่ประกอบด้วยตัวแปร Song s และ struct listnode *next ที่ชี้ไปยัง node ถัดไป จากนั้นทำการ typedef struct listnode LN; หลังจากนั้นจะนำค่าเวลาทั้งหมดมาหาผลรวมและแสดงค่าออกทางหน้าจอโปรแกรม

ภายในโปรแกรมจะมีฟังก์ชัน main () และฟังก์ชันย่อยที่ทำหน้าที่แตกต่างกันออกไปอยู่อีก 4 ฟังก์ชัน โดยแต่ละฟังก์ชันจะทำงานดังนี้

ฟังก์ชัน main () เป็นฟังก์ชันหลักที่ทำหน้าที่การรับค่าของเพลงและจำนวนเวลาของเพลงและแสดงค่าของผลรวมออกทางหน้าจอ ภายในฟังก์ชันมีการประกาศตัวแปร Song buf เพื่อเก็บค่าที่รับมาจากผู้ใช้ , LN *head=NULL เพื่อเป็น node ตัวแรกก่อนจะทำการใส่ node ใหม่ และ int i เพื่อนำมาวนลูป โปรแกรมจะเริ่มจากการวน loop for เงื่อนไข i=0; i<5; i++ ภายใน for loop จะมีการรับค่าชื่อเพลงในตัวแปร buf.title และเวลาของเพลงในตัวแปร buf.duration จากนั้นเรียกใช้ฟังก์ชัน insert(&head,&buf); โดยการส่ง address ของ head และ buf ไป เมื่อครบเงื่อนไข loop จะทำการเรียกใช้ฟังก์ชันแสดงค่า print(head); ซึ่งจะส่งค่าของ head ไป ต่อมาทำการเรียกใช้ฟังก์ชันผลรวมเวลา sum_duration(head) โดยส่งค่าของ head ไป และแสดงค่าที่รับคืนมาจากฟังก์ชันผลรวมเวลาออกมาทางหน้าจอ

ฟังก์ชันเพิ่ม node (void insert(LN **hptr , Song *sp)) เป็นฟังก์ชันที่จะทำการเพิ่ม node ของ link list แบบ insert at back โดยจะมีการประกาศตัวแปร pointer LN *new_node เพื่อเป็น node ใหม่ของ link list และตัวแปร pointer LN *tail เพื่อเป็นท้ายของ nodeเก่า การทำงานจะเริ่มจากการขอ memory ขณะโปรแกรมทำงาน malloc และให้ค่าของ new_node->s = *sp ที่รับเข้ามา และให้ new_node->next = NULL เพื่อให้ไปต่อตัวข้างหลังของ nodeเก่า จากนั้นทำการเรียกใช้ฟังก์ชันค้นหาท้าย tail = find_tail(*hptr); โดยการส่งค่าของ *hptr ไปและคืนค่ามาจากฟังก์ชันเก็บในตัวแปร tail ต่อมาใช้เงื่อนไข if (*hptr==NULL) ซึ่งจะให้ค่าของ *hptr=new_node เพื่อให้ node ตัวแรกที่เพิ่มเข้ามาใหม่เงื่อนไข else ให้ค่าของ tail->next =new_node เพื่อให้ node ที่เพิ่มมาไปต่อท้าย node แรก

ฟังก์ชันค้นหาท้าย node (LN *find_tail(LN *head)) เป็นฟังก์ชันที่ทำการค้นหา node ตัวสุดท้าย และคืนค่ากลับให้ฟังก์ชันเพิ่ม node (void insert(LN **hptr , Song *sp)) ภายในฟังก์ชันจะมีการประกาศตัวแปร pointer LN * tails เพื่อรับค่า address ของ node ตัวสุดท้าย จะเริ่มจากการใช้เงื่อนไข if (head==NULL) เพื่อเช็คว่าเป็น node ตัวแรกหรือไม่ ถ้าใช่ให้คืนค่าของ NULL เมื่อไม่ตรงเงื่อนไข if ให้ tails = head และทำการวน loop while เงื่อนไข tails->next != NULL โดยเงื่อนไข loop จะวนไปเรื่อยๆจนถึง node ตัวสุดท้าย ทำให้ได้ค่าของ address ของ node ตัวสุดท้ายมา ภายใน while loop จะใช้คำสั่ง tails = tails->next เพื่อเลื่อน node ไปเรื่อย ๆ เมื่อครบเงื่อนไข loop จะทำการคืนค่า tails กลับให้ฟังก์ชันเพิ่ม node (void insert(LN **hptr, Song *sp))

ฟังก์ชัน void print(LN *head) เป็นฟังก์ชันที่ทำหน้าที่ในการแสดงค่าชื่อเพลง title ที่บันทึกไว้ในแต่ละ node ออกมาใช้ โดยการใช้ประโยชน์จากการเชื่อมกัน ซึ่ง node ที่เก็บค่าตัวเลขที่รับมาแรกสุดจะแสดงค่าออกมาก่อน ในการแสดงค่าจะใช้การวน loop while เงื่อนไข คือ head != NULL ใน loop จะแสดงค่าของ head->s.title ในแต่ละ node ออกมาและให้ head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวนลูปเลื่อน node ไปเรื่อย ๆ จนกว่าเจอ NULL

ฟังก์ชัน double sum_duration(LN *head) จะมีประกาศตัวแปร double ssum = 0 จากนั้นวน loop เงื่อนไข while (head != NULL) ใน loop จะใช้คำสั่ง ssum += head->s.duration; เพื่อหาผลรวมของ duration ในแต่ละ node และเก็บค่าไว้ในตัวแปร ssum ต่อมาใช้คำสั่ง head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวน loop จนกว่าจะเจอ NULL หลังจากนั้นคืนค่า ssum ให้กับฟังก์ชัน main ()

ข้อที่ 2 : Linked Song 2

Linked Song 2

รับข้อมูล Song จากผู้ใช้จำนวน 5 เพลง เก็บไว้ในลิงคิสต์ โดยใช้เทคนิค insert at back โดยให้ถือว่าชื่อเพลงไม่มีเว้นวรรค เขียนโค้ดโดยใช้ต้นแบบของฟังก์ชันที่ให้มาอย่างเหมาะสม

```
S01: <I_Love_You 180.4>
...
= 3120.50
```

Code

lab07-2.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct song
4  {
5      char title[128];
6      double duration;
7  }; typedef struct song Song;
8
9  struct listnode
10 {
11     Song *sp;
12     struct listnode *next;
13 }; typedef struct listnode LN;
14
15 Song *input_song();
16 void insert(LN **hptr, Song *sp);
17 LN *find_tail(LN *head);
18 double sum_duration(LN *head);
19 void print(LN *head);
20
21 int main()
22 {
23     LN *head=NULL;
24     Song *sp;
25     int i;
26     for(i = 0; i < 5; i++)
27     {
28         printf("Song[%0d] : ",i+1);
29         sp = input_song(); //malloc, scanf Song in this function
30         insert(&head,sp);
31     }
32     printf("\n");
33     print(head);
34     printf("\nTotal duration = %.2lf",sum_duration(head));
35     return 0;
36 }
```

```

37 Song *input_song()
38 {
39     Song *s;
40     s = (Song*)malloc(sizeof(Song));
41     fflush(stdin);
42     scanf("%[^\n]s", s->title);
43     printf("Duration : ");
44     scanf("%lf", &s->duration);
45     return s;
46 }
47
48 void insert(LN **hptr, Song *sp)
49 {
50     LN *new_node, *tail;
51     new_node = (LN*)malloc(sizeof(LN));
52     new_node->sp = sp;
53     new_node->next = NULL;
54
55     tail = find_tail(*hptr);
56     if(*hptr == NULL)
57     {
58         *hptr = new_node;
59     }
60     else
61     {
62         tail->next = new_node;
63     }
64 }
65
66

```

```

67 LN *find_tail(LN *head)
68 {
69     LN *tails;
70     if(head == NULL)
71     {
72         return NULL;
73     }
74     else
75     {
76         tails = head;
77         while(tails->next != NULL)
78         {
79             tails = tails->next;
80         }
81         return tails;
82     }
83 }
84

```

```

85 void print(LN *head)
86 {
87     int i=1;
88     while (head != NULL)
89     {
90         printf("s%d:%s ", i, head->sp->title);
91         head = head->next;
92         i++;
93     }
94 }
95
96 double sum_duration(LN *head)
97 {
98     double ssum=0;
99     while (head != NULL)
100     {
101         ssum += head->sp->duration;
102         head = head->next;
103     }
104     return ssum;
105 }

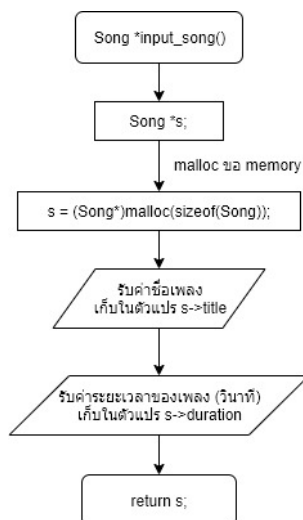
```

Flowchart

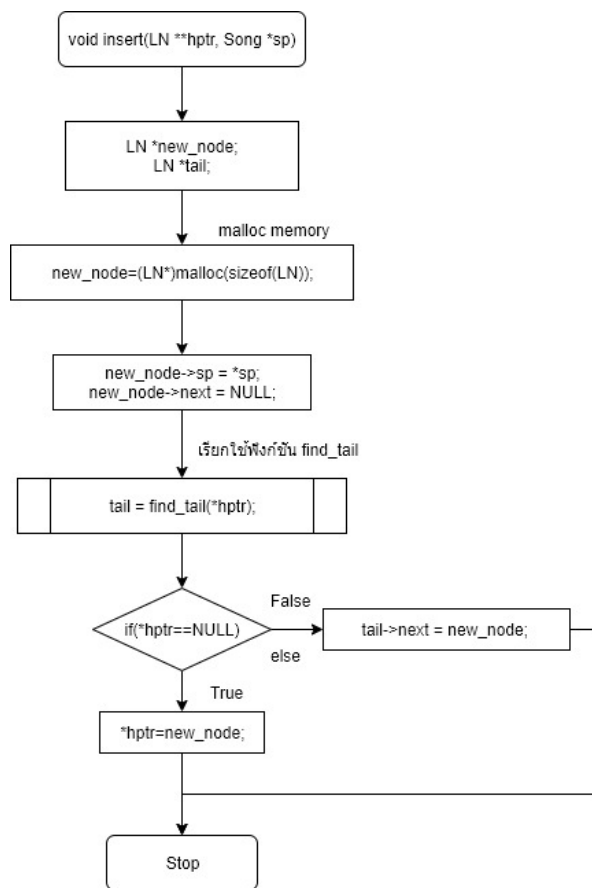
ฟังก์ชัน main()



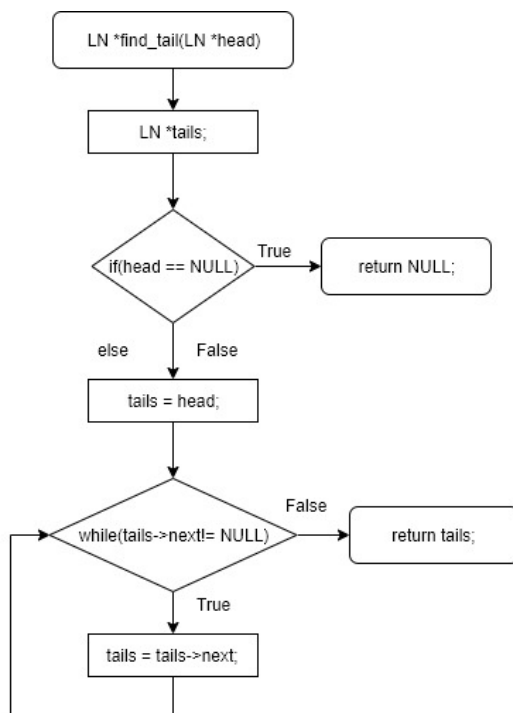
ฟังก์ชัน Song *input_song()



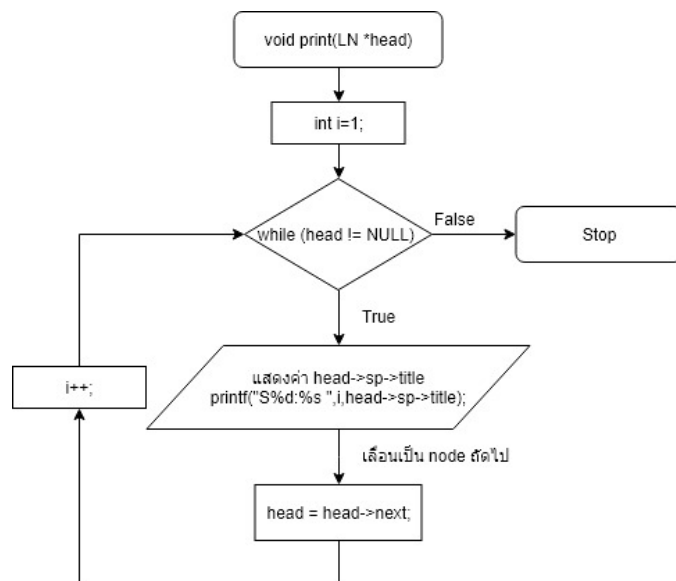
ฟังก์ชัน void insert(LN **hptr, Song *sp)



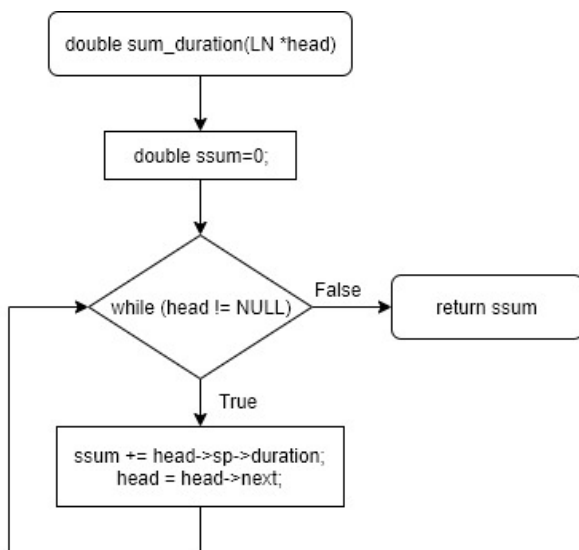
ฟังก์ชัน LN *find_tail(LN *head)



ฟังก์ชัน void print(LN *head)



ฟังก์ชัน double sum_duration(LN *head)



ผลการรันโปรแกรม

```
C:\Users\User\OneDrive\Programming & Data\lab07\lab07-2.exe
Song[1] : Firework
Duration : 233
Song[2] : Blank_Space
Duration : 272
Song[3] : One_Call_Away
Duration : 241
Song[4] : Numb
Duration : 187
Song[5] : 7_Years
Duration : 240

S1:Firework S2:Blank_Space S3:One_Call_Away S4:Numb S5:7_Years
Total duration = 1173.00
-----
Process exited after 187.5 seconds with return value 0
Press any key to continue . . . _
```

อธิบายหลักการทำงาน : เป็นโปรแกรมที่รับค่าข้อมูลเพลง 5 จำนวนที่ประกอบไปด้วยชื่อและจำนวนเวลาหน่วยเป็นวินาที โดยใช้ link list แบบ insert at back โดยสร้างโครงสร้างข้อมูล Struct song ภายในโครงสร้างจะมีตัวแปรเก็บค่าชื่อเพลง char title[128] และตัวแปรเก็บค่าจำนวนเวลา double duration; จากนั้น typedef struct song Song; ต่อมาสร้างโครงสร้างข้อมูล struct listnode ที่ประกอบด้วยตัวแปร Song *sp และ struct listnode *next ที่ชี้ไปยัง node ถัดไป จากนั้นทำการ typedef struct listnode LN; หลังจากนั้นจะนำค่าเวลาทั้งหมดมาหาผลรวมและแสดงค่าออกทางหน้าจอโปรแกรม

ภายในโปรแกรมจะมีฟังก์ชัน main () และฟังก์ชันย่อยที่ทำหน้าที่แตกต่างกันออกไปอยู่อีก 5 ฟังก์ชัน โดยแต่ละฟังก์ชันจะทำงานดังนี้

ฟังก์ชัน main () เป็นฟังก์ชันหลักทำหน้าที่การรับค่าของเพลงและจำนวนเวลาของเพลงและแสดงค่าของผลรวมออกทางหน้าจอ ภายในฟังก์ชันจะมีการประกาศตัวแปร Song *sp เพื่อเก็บค่าที่รับมาจากผู้ใช้ , LN *head=NULL เพื่อเป็น node ตัวแรกก่อนจะทำการใส่ node ใหม่ และ int i เพื่อนำมาวนลูปโปรแกรมจะเริ่มจากการวน loop for เงื่อนไข i=0; i<5; i++ ภายใน for loop จะมีการเรียกใช้ฟังก์ชันรับค่า sp = input_song(); โดยให้ค่า pointer sp เท่ากับค่าที่คืนมาจากฟังก์ชันนี้ และเรียกใช้ฟังก์ชันเพิ่ม node insert(&head,sp); ซึ่งส่งค่า address ของ head และค่าของ sp ไป เมื่อ loop ครบเงื่อนไขจะทำการเรียกใช้ฟังก์ชันแสดงค่า print(head); โดยการส่งค่าของ head ไป ต่อมาทำการเรียกใช้ฟังก์ชันผลรวมเวลา sum_duration(head) โดยส่งค่าของ head ไป และแสดงค่าที่รับคืนมาจากฟังก์ชันผลรวมเวลาออกทางหน้าจอ

ฟังก์ชันรับค่า (Song * input()) เป็นฟังก์ชันที่จะรับค่าจากผู้ใช้ในการประกาศตัวแปร pointer Song *s แล้วทำการ malloc ขอ memory ขณะโปรแกรมทำงาน จากนั้นรับค่าชื่อเพลงเก็บค่าในตัวแปร s->title และรับค่าเวลาของเพลงเก็บค่าในตัวแปร s->duration ต่อมาทำการคืนค่าของ s ให้ฟังก์ชัน main()

ฟังก์ชันเพิ่ม node (void insert(LN **hptr , Song *sp)) เป็นฟังก์ชันที่จะทำการเพิ่ม node ของ link list แบบ insert at back โดยจะมีการประกาศตัวแปร pointer LN *new_node เพื่อเป็น node ใหม่ของ link list และตัวแปร pointer LN *tail เพื่อเป็นท้ายของ nodeเก่า การทำงานจะเริ่มจากการขอ memory ขณะโปรแกรมทำงาน malloc และให้ค่าของ new_node->sp = *sp ที่รับเข้ามา และให้ new_node->next = NULL เพื่อให้ไปต่อตัวข้างหลังของ nodeเก่า จากนั้นทำการเรียกใช้ฟังก์ชันค้นหาท้าย tail = find_tail(*hptr); โดยการส่งค่าของ *hptr ไปและคืนค่ามาจากฟังก์ชันเก็บในตัวแปร tail ต่อมาใช้เงื่อนไข if (*hptr==NULL) ซึ่งจะให้ค่าของ *hptr=new_node เพื่อให้ node ตัวแรกที่เพิ่มเข้ามาใหม่เงื่อนไข else ให้ค่าของ tail->next =new_node เพื่อให้ node ที่เพิ่มมาไปต่อท้าย node แรก

ฟังก์ชันค้นหาท้าย node (LN *find_tail(LN *head)) เป็นฟังก์ชันที่ทำการค้นหา node ตัวสุดท้าย และคืนค่ากลับให้ฟังก์ชันเพิ่ม node (void insert(LN **hptr , Song *sp)) ภายในฟังก์ชันจะมีการประกาศตัวแปร pointer LN * tails เพื่อรับค่า address ของ node ตัวสุดท้าย จะเริ่มจากการใช้เงื่อนไข if (head==NULL) เพื่อเช็คว่าเป็น node ตัวแรกหรือไม่ ถ้าใช่ให้คืนค่าของ NULL เมื่อไม่ตรงเงื่อนไข ให้ tails = head และทำการวน loop while เงื่อนไข tails->next != NULL โดยเงื่อนไข loop จะวนไปเรื่อยๆถึง node ตัวสุดท้าย ทำให้ได้ค่าของ address ของ node ตัวสุดท้ายมา ภายใน while loop จะใช้คำสั่ง tails = tails->next เพื่อเลื่อน node ไปเรื่อย ๆ เมื่อครบเงื่อนไข loop จะทำการคืนค่า tails กลับให้ฟังก์ชันเพิ่ม node (void insert(LN **hptr, Song *sp))

ฟังก์ชัน void print(LN *head) เป็นฟังก์ชันที่ทำหน้าที่ในการแสดงค่าชื่อเพลง title ที่บันทึกไว้ในแต่ละ node ออกมาใช้ โดยการใช้ประโยชน์จากการเชื่อมกัน ซึ่ง node ที่เก็บค่าตัวเลขที่รับมาแรกสุดจะแสดงค่าออกมาก่อน ในการแสดงค่าจะใช้การวน loop while เงื่อนไข คือ head != NULL ใน loop จะแสดงค่าของ head->sp->title ในแต่ละ node ออกมาและให้ head = head->next เพื่อเลื่อนไปยัง node ถัดไป และวนลูปเลื่อน node ไปเรื่อย ๆ จนกว่าเจอ NULL

ฟังก์ชัน `double sum_duration(LN *head)` จะมีประกาศตัวแปร `double ssum = 0` จากนั้นวน loop เงื่อนไข `while (head != NULL)` ใน loop จะใช้คำสั่ง `ssum += head->sp->duration;` เพื่อหาผลรวมของ duration ในแต่ละ node และเก็บค่าไว้ในตัวแปร ssum ต่อมาใช้คำสั่ง `head = head->next` เพื่อเลื่อนไปยัง node ถัดไป และวน loop จนกว่าจะเจอ NULL หลังจากนั้นคืนค่า ssum ให้กับฟังก์ชัน `main ()`

ความรู้จากการทำ Lab Linked Song : การค้นหาท้ายของ node โดยการวน loop เลื่อน node ไปเรื่อย ๆ จนกว่าจะเจอ NULL เพื่อนำ node ที่เก็บข้อมูลที่ได้รับค่าข้อมูลมาใหม่ มาต่อท้าย node ตัวถัดไป ซึ่งเป็นการเพิ่มข้อมูลในรูปแบบ insert at back