รายงาน : Lab07 – More Linked List

จัดทำโดย

นายกิตติพศ หนูทอง รหัสนักศึกษา : 6135512003

นายปฏิภาณ วรรณโก รหัสนักศึกษา : 6135512059

Section : 01

240-207 Programing and Data Structures

“งานทั้งหมดนี้ในรายงานฉบับนี้ล้วนเป็นผลงานของข้าพเจ้า มิได้ลอกหรือสำเนามาจากที่อื่นใดในกรณีที่พบว่าเกิดสำเนาด้วยวิธีใดก็ตาม ข้าพเจ้ายินดีไม่ขอรับคะแนนจากรายงานฉบับนี้”

คะแนนที่ได้

ลงชื่อ

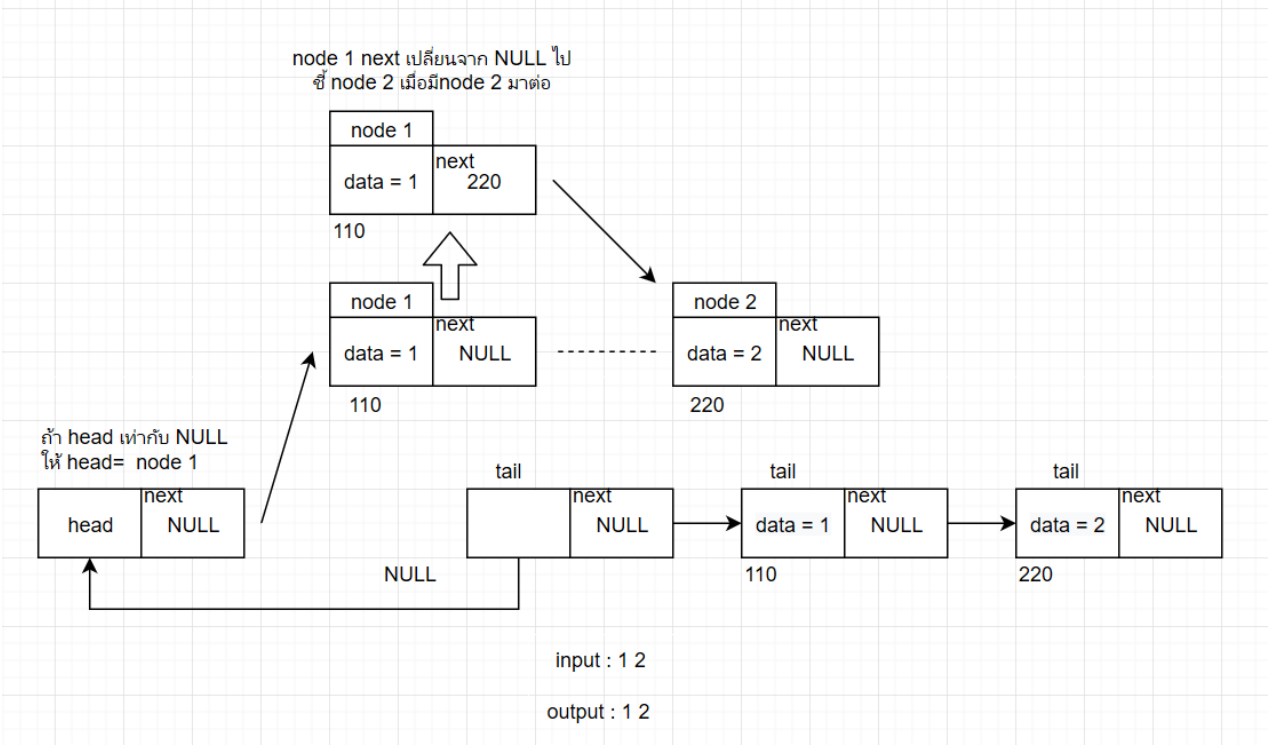
กิตติพศ หนูทอง ปฏิภาณ วรรณโก

(นายกิตติพศ หนูทอง) (นายปฏิภาณ วรรณโก)

Lab07 – More Linked List

แผนภาพ Linked List แบบ Insert At Back

แนวคิด : เป็นการเลื่อนหา node ท้ายของ Linked List เพื่อนำ node ที่เก็บค่าข้อมูลที่เพิ่มมาใหม่ไปต่อท้ายเป็น node ถัดไป



ข้อที่ 1 : Linked Song 1

Linked Song 1

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

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

Code

lab07-1.c

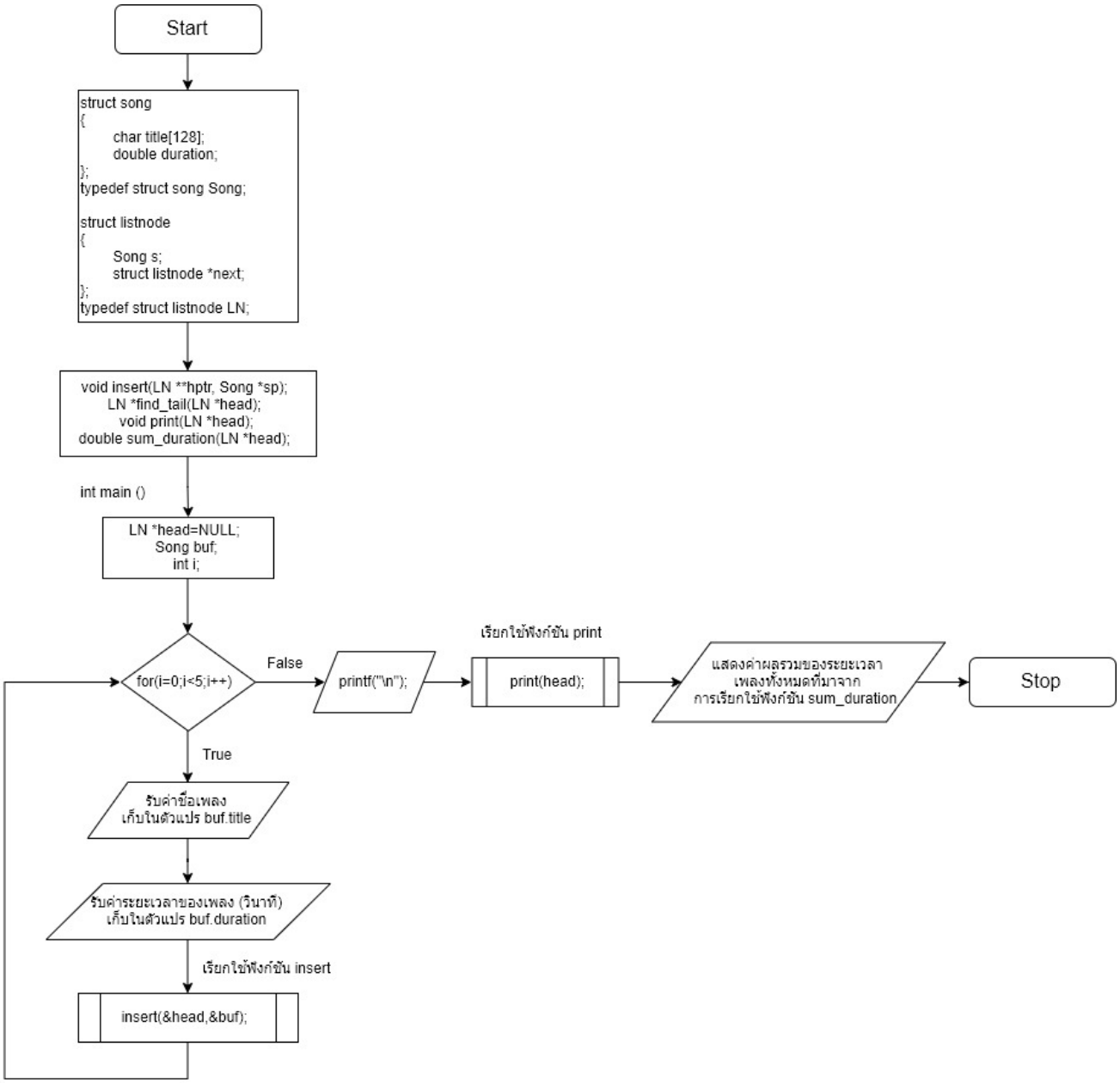
```c
#include<stdio.h>
#include<stdlib.h>
struct song
{
    char title[128];
    double duration;
};
typedef struct song Song;

struct listnode
{
    Song s;
    struct listnode *next;
};
typedef struct listnode LN;

void insert(LN **hptr, Song *sp);
LN *find_tail(LN *head);
void print(LN *head);
double sum_duration(LN *head);

int main()
{
    LN *head=NULL;
    Song buf;
    int i;
    for(i=0;i<5;i++)
    {
        fflush(stdin);
        printf("Song[%0d] : ",i+1);
        scanf("%[^\n]s",buf.title);
        printf("Duration : ");
        scanf("%lf",&buf.duration);
        insert(&head,&buf);
    }
    printf("\n");
    print(head);
    printf("\nTotal duration = %.2lf",sum_duration(head));
    return 0;
}
```

```c
void insert(LN **hptr, Song *sp)
{
    LN *new_node;
    LN *tail;
    new_node=(LN*)malloc(sizeof(LN));
    new_node->s = *sp;
    new_node->next = NULL;

    tail = find_tail(*hptr);
     if(*hptr==NULL)
    {
        *hptr=new_node;
    }
    else
    {
        tail->next = new_node;
    }
}

LN *find_tail(LN *head)
{
    LN *tails;
     if(head == NULL)
    {
        return NULL;
    }
    else
    {
        tails = head;
        while(tails->next!= NULL)
        {
            tails = tails->next;
        }
        return tails;
    }
}
```

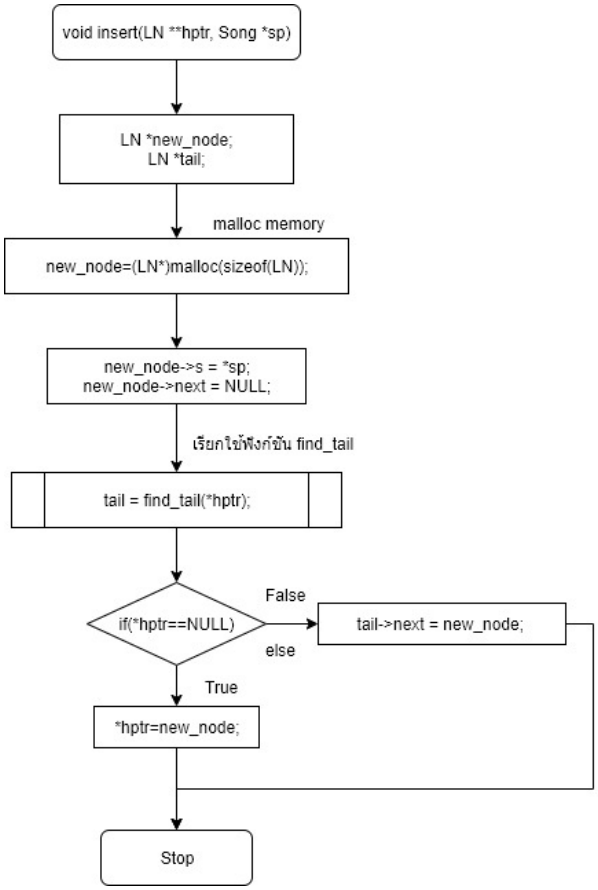
```c
void print(LN *head)
{
    int i=1;
    while (head != NULL)
    {
        printf("S%d:%s ",i,head->s.title);
        head = head->next;
        i++;
    }
}

double sum_duration(LN *head)
{
    double ssum=0;
    while (head != NULL)
    {
        ssum += head->s.duration;
        head = head->next;
    }
    return ssum;
}
```

Flowchart

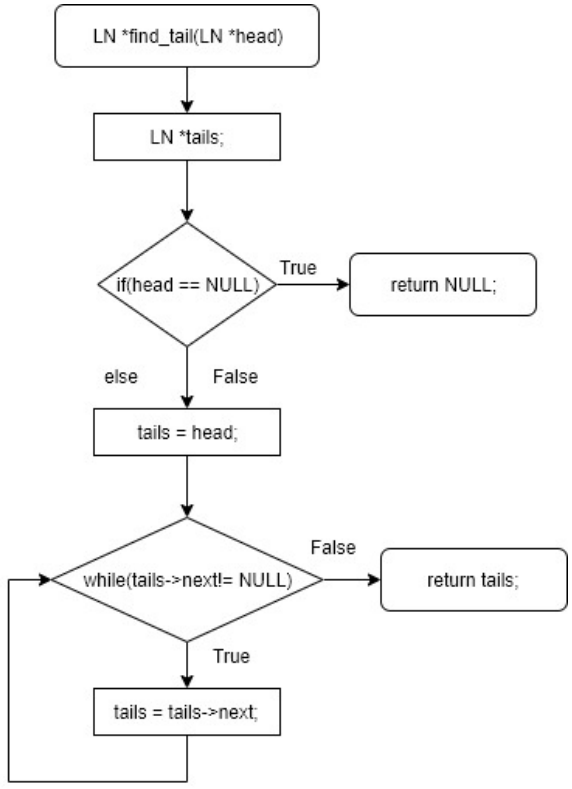
ฟังก์ชัน main()



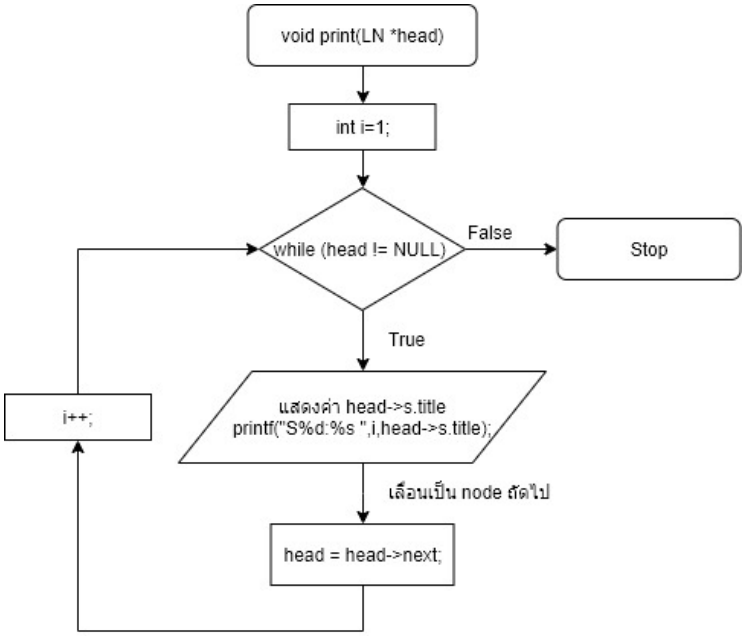
ฟังก์ชัน void insert(LN **hptr, Song *sp)



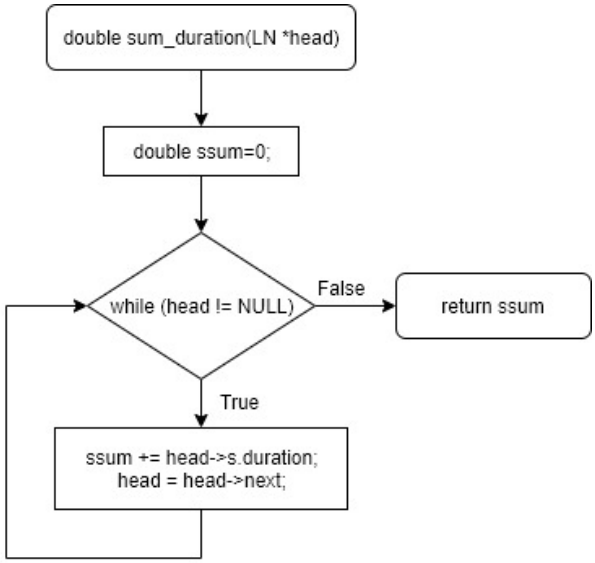
ฟังก์ชัน LN *find_tail(LN *head)



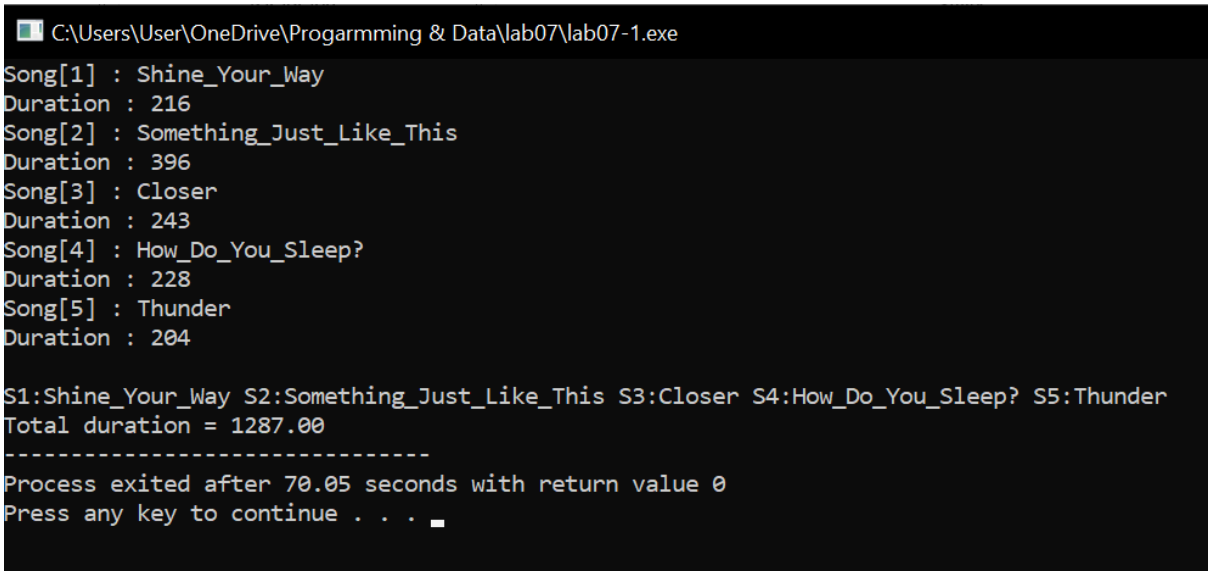
ฟังก์ชัน void print(LN *head)



ฟังก์ชัน double sum_duration(LN *head)



ผลการรันโปรแกรม



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 node เก่า การทำงานจะเริ่มจากการขอ memory ขณะโปรแกรมทำงาน malloc และให้ค่าของ new_node->s = *sp ที่รับเข้ามา และให้ new_node->next = NULL เพื่อให้ไปต่อตัวข้างหลังของ node เก่า จากนั้นทำการเรียกใช้ฟังก์ชันค้าหาท้าย tail = find_tail(*hptr); โดยการส่งค่าของ *hptr ไปและคืนค่ามาจากฟังก์ชันเก็บในตัวแปร tail ต่อมาใช้เงื่อนไข if (*hptr==NULL) ซึ่งจะให้ค่าของ *hptr=new_node เพื่อให้ node ตัวแรกที่เพิ่มเข้ามาใหม่ เงื่อนไข else ให้ค่าของ tail->next =new_node เพื่อให้ node ที่เพิ่มมาไปต่อท้าย node แรก

ฟังก์ชันค้าหาท้าย node (LN *find_tail(LN *head)) เป็นฟังก์ชันที่ทำการค้นหาnodeตัวทุดท้าย 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 เก็บไว้ในลิงค์ลิสต์ โดยใช้เทคนิค insert at back โดยให้ถือว่าชื่อเพลงไม่มีเว้นวรรค เขียนโค้ดโดยใช้ต้นแบบของฟังก์ชันที่ให้มาอย่างเหมาะสม

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

Code

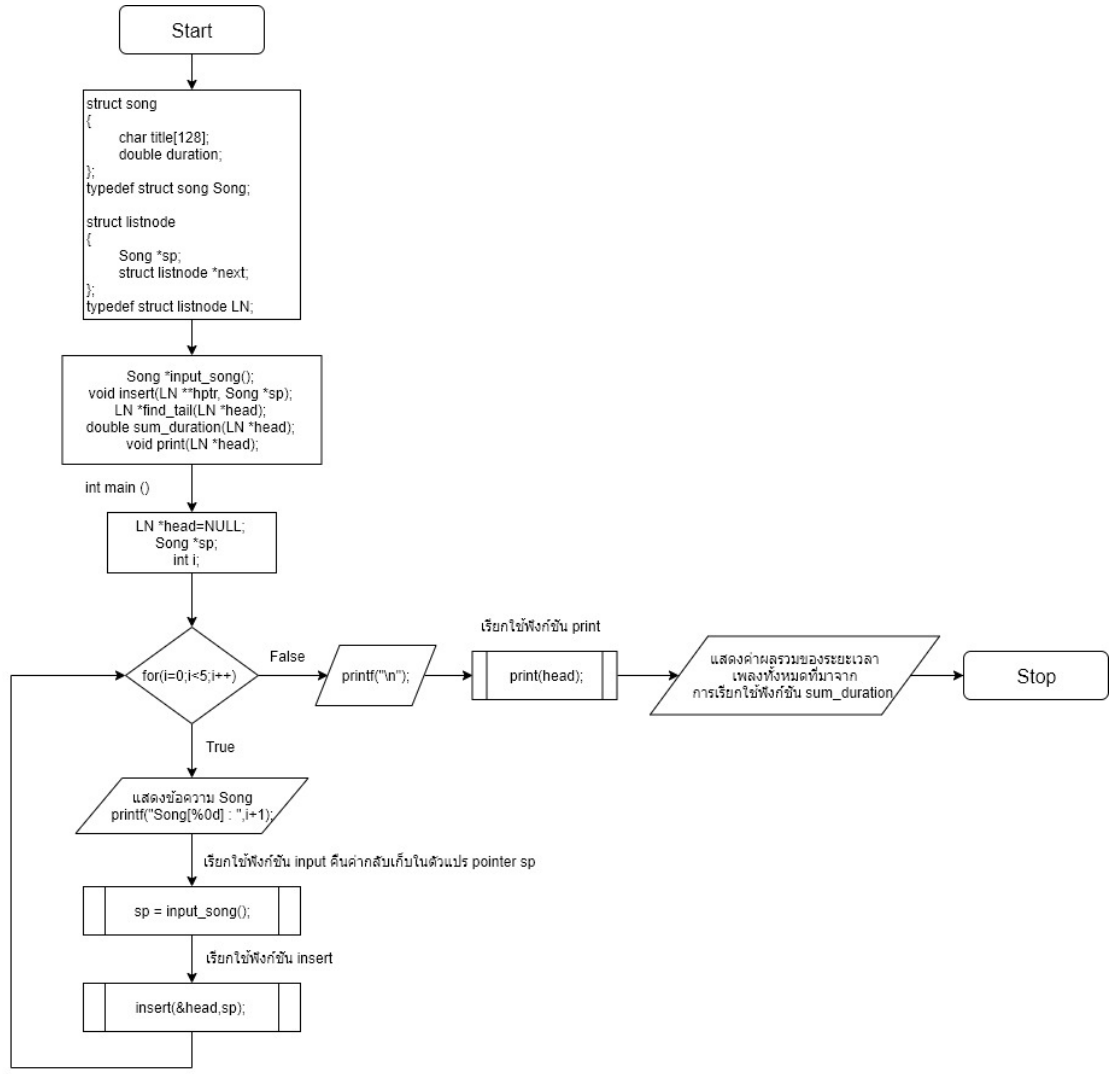
lab07-2.c

```c
#include<stdio.h>
#include<stdlib.h>
struct song
{
    char title[128];
    double duration;
}; typedef struct song Song;

struct listnode
{
    Song *sp;
    struct listnode *next;
}; typedef struct listnode LN;

Song *input_song();
void insert(LN **hptr, Song *sp);
LN *find_tail(LN *head);
double sum_duration(LN *head);
void print(LN *head);

int main()
{
    LN *head=NULL;
    Song *sp;
    int i;
    for(i = 0; i < 5; i++)
    {
        printf("Song[%0d] : ",i+1);
        sp = input_song();  //malloc, scanf Song in this function
        insert(&head,sp);
    }
    printf("\n");
    print(head);
    printf("\nTotal duration = %.2lf",sum_duration(head));
    return 0;
}
```

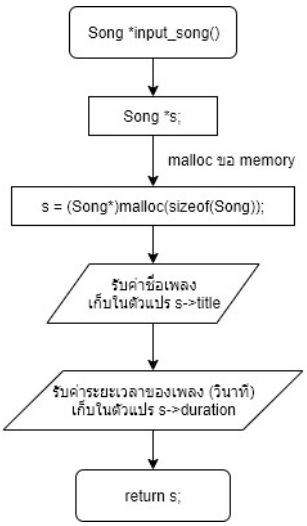
```c
Song *input_song()
{
    Song *s;
    s = (Song*)malloc(sizeof(Song));
    fflush(stdin);
    scanf("%[^\n]s",s->title);
    printf("Duration : ");
    scanf("%lf",&s->duration);
    return s;
}

void insert(LN **hptr, Song *sp)
{
    LN *new_node,*tail;
    new_node=(LN*)malloc(sizeof(LN));
    new_node->sp = sp;
    new_node->next = NULL;

    tail = find_tail(*hptr);
     if(*hptr==NULL)
    {
        *hptr=new_node;
    }
    else
    {
        tail->next = new_node;
    }

}

LN *find_tail(LN *head)
{
     LN *tails;
      if(head == NULL)
    {
        return NULL;
    }
    else
    {
        tails = head;
        while(tails->next!= NULL)
        {
            tails = tails->next;
        }
        return tails;
    }
}

void print(LN *head)
{
    int i=1;
    while (head != NULL)
    {
        printf("S%d:%s ",i,head->sp->title);
        head = head->next;
        i++;
    }
}

double sum_duration(LN *head)
{
    double ssum=0;
    while (head != NULL)
    {
        ssum += head->sp->duration;
        head = head->next;
    }
    return ssum;
}
```

Flowchart

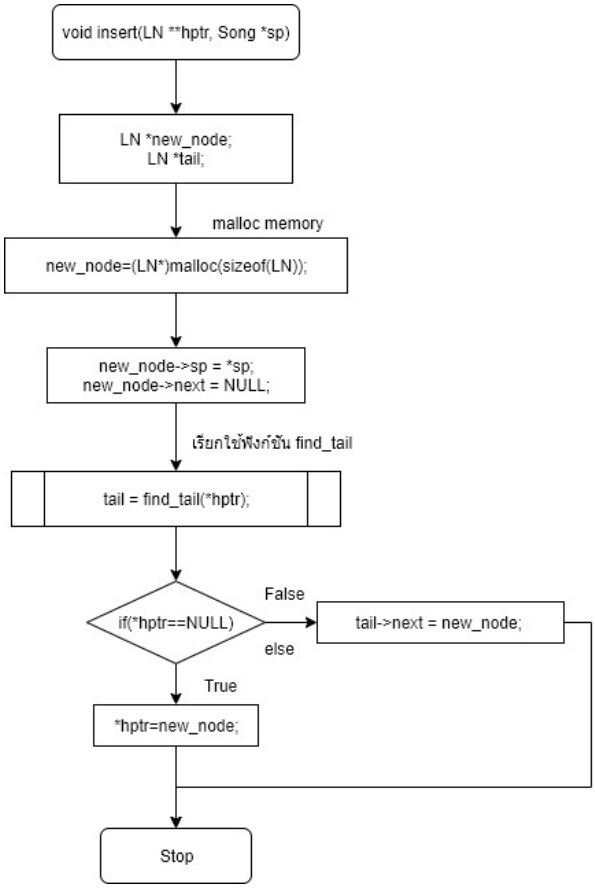
ฟังก์ชัน main()



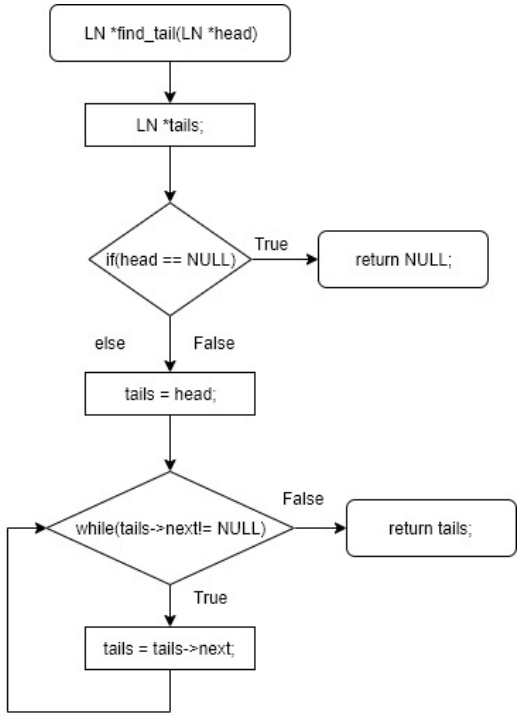
ฟังก์ชัน Song *input_song()



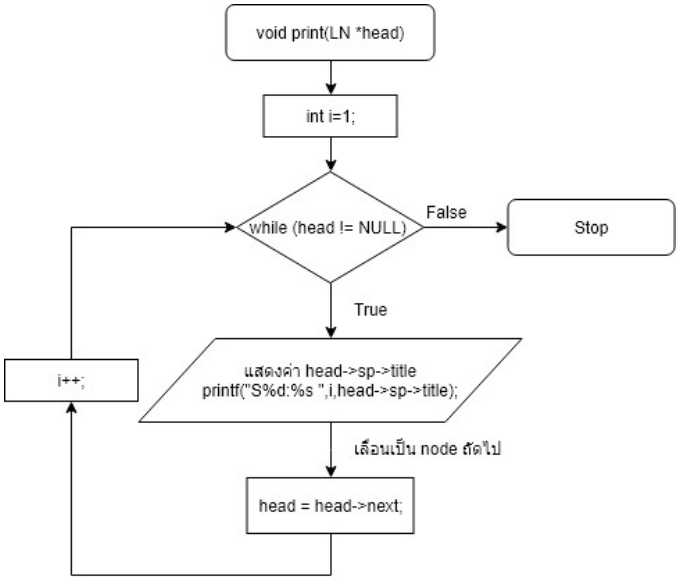
ฟังก์ชัน void insert(LN **hptr, Song *sp)



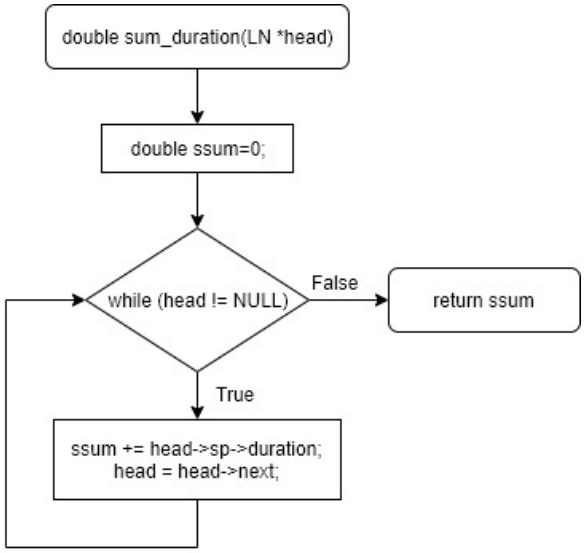
ฟังก์ชัน LN *find_tail(LN *head)



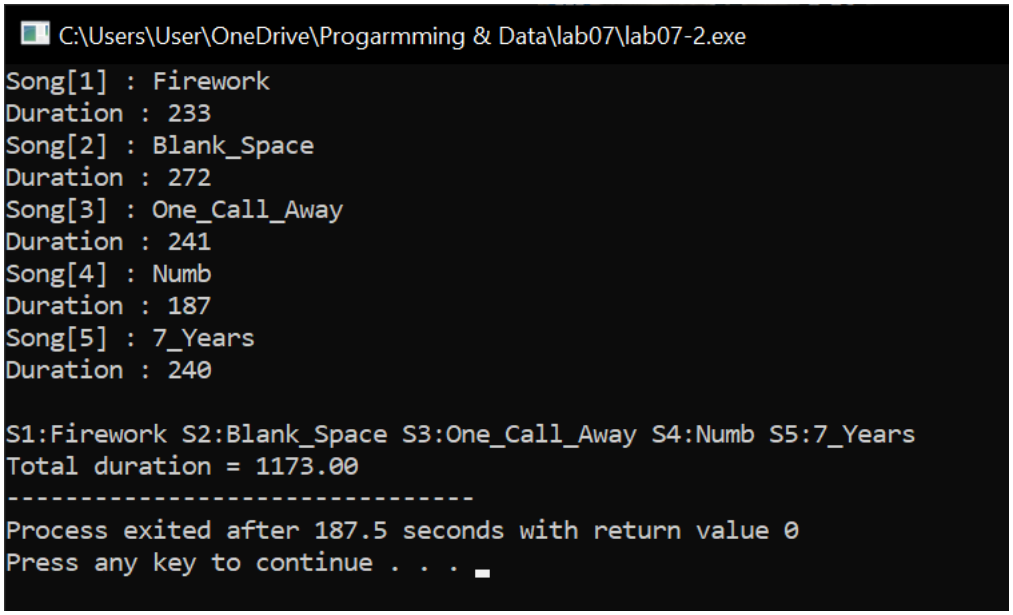
ฟังก์ชัน void print(LN *head)



ฟังก์ชัน double sum_duration(LN *head)



ผลการรันโปรแกรม



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

ภายในโปรแกรมจะมีฟังก์ชันเมน 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 และแสดงค่าที่รับคืนมาจากฟังก์ชันผลรวมเวลาออกมาทางหน้าจอ

ฟังก์ชันรับค่า (Song * input()) เป็นฟังก์ชันที่จะรับค่าจากผู้ใช้โดยการประกาศตัวแปร 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 node เก่า การทำงานจะเริ่มจากการขอ memory ขณะโปรแกรมทำงาน malloc และให้ค่าของ new_node->sp = *sp ที่รับเข้ามา และให้ new_node->next = NULL เพื่อให้ไปต่อตัวข้างหลังของ node เก่า จากนั้นทำการเรียกใช้ฟังก์ชันค้าหาท้าย tail = find_tail(*hptr); โดยการส่งค่าของ *hptr ไปและคืนค่ามาจากฟังก์ชันเก็บในตัวแปร tail ต่อมาใช้เงื่อนไข if (*hptr==NULL) ซึ่งจะให้ค่าของ *hptr=new_node เพื่อให้ node ตัวแรกที่เพิ่มเข้ามาใหม่ เงื่อนไข else ให้ค่าของ tail->next =new_node เพื่อให้ node ที่เพิ่มมาไปต่อท้าย node แรก

ฟังก์ชันค้าหาท้าย node (LN *find_tail(LN *head)) เป็นฟังก์ชันที่ทำการค้นหาnodeตัวทุดท้าย 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->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 ที่เก็บข้อมูลที่รับค่าข้อมูลมาใหม่ มาต่อท้าย node ตัวถัดไป ซึ่งเป็นการเพิ่มข้อมูลในรูปแบบ insert at back