CP 111 Computer Programming, 2/2554, Section B01/B02

Programming Homework 11

กำหนดส่งภายในวันจันทร์ที่ 19 ก.พ. 2555 เวลา 23.59 น.

คำเตือน

นิสิตสามารถจับกลุ่มปรึกษาหารือกันได้ แต่นิสิตแต่ละคนในกลุ่มจะต้องลงมือเขียนโปรแกรมด้วยตนเองและส่งโปรแกรมของตนเองโดยไม่ขึ้นอยู่กับสมาชิกในกลุ่ม ***และอย่านำโปรแกรมของตนเองให้กับผู้อื่นดู***

ถ้าตรวจพบว่านิสิตนำงานของคนอื่นมาส่งหรือนิสิตก็อปปี้ แก้ไข ดัดแปลงโปรแกรมของผู้อื่น จะมีโทษคือปรับตก (ได้เกรด E) ในวิชานี้ทันที

หมายเหตุ

1. ข้อกำหนดการตั้งชื่อไฟล์การบ้านคือ xxx_**HW**yy_**Q**zz.* โดยที่ xxx = รหัสสามตัวท้ายของเลขประจำตัว, yy = หมายเลข homework, zz = หมายเลขข้อในแต่ละ homework (ตัวอย่างเช่น ไฟล์โปรแกรม ข้อ 1 การบ้านครั้งที่ 1 ของนิสิตรหัส 000 คือ 000_HW01_Q01.c) และในแต่ละข้อ

2. การส่งการบ้าน ให้ส่งไฟล์โปรแกรมไปตรวจกับระบบ Café-grader ที่ http://cslab.swu.ac.th/grader

3. การเขียนโปรแกรมที่คำนวณกับตัวเลขที่มีจุดทศนิยม ให้ประกาศตัวแปรที่เกี่ยวข้องเป็นแบบ `double` (Double-precision floating point)

1. Problem name: `Linked_List_Add_Student_Ascend`

กำหนดให้นิยามของโครงสร้างของ node หนึ่งๆ ใน linked list เป็นดังข้างล่าง

```
typedef struct s_node_tag S_NODE;
struct  s_node_tag {
  int ID;
  char name[21];
  int score;
  S_NODE *next;
};
```

จงเขียนโปรแกรม (ทำโปรแกรมข้างล่างให้สมบูรณ์) เพื่ออ่านข้อมูลของนักเรียนที่ลงทะเบียนเรียนในวิชาหนึ่งๆ มาเก็บไว้ใน Linked list โดยการเพิ่ม node ใหม่ลงใน linked list ให้นำไปใส่ในลำดับที่ทำให้ node เรียงต่อกันตามค่า ID จากน้อยไปมาก กำหนดให้เขียนโค้ดในใช้ฟังก์ชัน `Add_Student_Ascend`

หลังจากนั้นให้แสดงข้อมูลของนักเรียนที่เก็บอยู่ใน linked list เรียงจาก node แรก (head node) ไปยัง node สุดท้าย (tail node)

นอกจากนี้ให้เขียนโค้ดในฟังก์ชัน `Print_All_Students` ซึ่งจะทำการแสดงข้อมูลของทุกโหนด (รหัส ชื่อ คะแนนของนักเรียนทุกคน) ใน linked list และเขียนโค้ดในฟังก์ชัน `Free_All_Students` ซึ่งจะทำการ free หน่วยความจำที่ได้จองไว้ตอนสร้าง Linked list โดยที่ฟังก์ชันนี้จะถูกเรียกตอนท้ายของโปรแกรมเพื่อทำการคืนหน่วยความที่ได้จองกลับสู่ระบบ

กำหนดให้ข้อมูลนำเข้าในแต่ละบรรทัด ประกอบด้วย รหัสนิสิต ชื่อไม่เกิน 20 ตัวอักษร คะแนนสอบ (เลขจำนวนเต็ม) เช่น 10101 John 60 เป็นต้นโดยในกรณีที่ข้อมูลบรรทัดสุดท้ายคือ -1 -1 -1 จะเป็นบรรทัดที่ระบุจุดสิ้นสุดของข้อมูล

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
10101 John 60 10009 Sarah 50 10001 Abby 100 10084 Onny 90 11101 Jane 15 12009 Soe 80 12001 Tab 80 13084 Anny 50 13184 Peter 40 13284 Bob 40 -1 -1 -1	110001 Abby 100 10009 Sarah 50 10084 Onny 90 10101 John 60 11101 Jane 15 12001 Tab 80 12009 Soe 80 13084 Anny 50 13184 Peter 40 13284 Bob 40

หมายเหตุ ผลลัพธ์ที่ข้อมูลส่งออกจะแสดงข้อมูลใน linked list จาก node แรกไปยัง node สุดท้าย ซึ่งจะเรียงตามลำดับของค่า ID จากน้อยไปมาก

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct  s_node_tag {
  int ID;
  char name[21];
  int score;
  S_NODE *next;
};

int Add_Student_Ascend(S_NODE *head,int ID,char *name,int score);
void Print_All_Students(S_NODE *head);
void Free_All_Students(S_NODE *head);

int main()
{
S_NODE dummy_head,*ptr,*pMin;
int s_id,s_score,min_score;
char s_name[21];

	dummy_head.next=NULL;
	while(1)
	{
		scanf("%d %s %d",&s_id,s_name,&s_score);
		if(s_id==-1)
			break;
		Add_Student_Ascend(&dummy_head,s_id,s_name,s_score);
	}

	Print_All_Students(&dummy_head);

	Free_All_Students(&dummy_head);

	return 0;
}


int Add_Student_Ascend(S_NODE *head,int id,char *name,int score)
{
S_NODE *new_s,*prev,*curr;

	// เซ็ต curr ให้ชี้ไปที่ node แรก และเซ็ต prev ให้ชี้ไปที่ node ก่อนหน้า curr
	prev=.....
	curr=.....
	// วนจนกว่าจะเจอ tail node (NULL)
	while(.....)
	{
		// วนหาโหนด curr โหนดแรกที่ ID มากกว่าหรือเท่ากับอาร์กิวเมนต์ id
		if(.....)
			break;
		// เปลี่ยนค่า prev และ curr ให้ชี้ไปที่โหนดถัดไป
		prev=.....
		curr=.....
	}
	// เพิ่มโหนดใหม่ new_s ให้อยู่ระหว่างโหนดที่ชี้โดย prev และ โหนดที่ชี้โดย curr
	new_s=.....
	if(.....)
		return -1;
```

```
        // กำหนดฟิลด์ที่เกี่ยวข้องกับข้อมูล (ID, name, score) ในโหนด

        new_s->ID=.......................................

        ..........................................................................

        ..........................................................................

        // กำหนดให้ prev ชี้ไปที่ โหนดใหม่ new_s และ
        // กำหนดให้ โหนดใหม่ new_s ชี้ไปที่โหนดที่ชี้โดย curr

        prev->next=.......................................

        ..........................................................................
        return 0;
}

void Free_All_Students(S_NODE *head)
{
S_NODE *del_node,*ptr;

        // เซ็ต ptr ให้ชี้ไปที่ node แรก
        ptr=...................................
        while(.......................................)
        {
                // กำหนดให้ del_node ให้ชี้ไปที่ node ที่ชี้โดย ptr
                del_node=..........................

                // เปลี่ยนค่า ptr ให้ชี้ไปที่โหนดถัดไป
                ........................................
                // คืนหน่วยความจำที่จองไว้สำหรับ node ที่ชี้โดย del_node
                ........................................
        }
}


void Print_All_Students(S_NODE *head)
{
S_NODE *ptr;

        // เซ็ต ptr ให้ชี้ไปที่ node แรก
        ptr=...................................
        while(.......................................)
        {
                // แสดงข้อมูลในโหนด ptr
                ........................................
                // เปลี่ยนค่า ptr ให้ชี้ไปที่โหนดถัดไป
                ........................................
        }
}
```

2. Problem name: `Linked_List_Is_Subset`

จงเขียนโปรแกรม ต่อจากโปรแกรมในข้อที่แล้ว) เพื่อทำการตรวจสอบว่า linked list จำนวน 2 linked list เป็น subset กันหรือไม่ <u>โดยที่ linked list หนึ่งๆ สมมติคือ linked list A จะเป็น subset ของอีก linked list หนึ่งสมมติคือ B ก็ต่อเมื่อทุกๆ โหนดใน linked list A จะต้องมีโหนดใน linked list B ที่ข้อมูล ID เหมือนกัน</u>

กำหนดให้เขียนและเรียกใช้ฟังก์ชัน `Linked_List_Is_Subset` ดังนิยามข้างล่าง ซึ่งจะทำการตรวจสอบว่า linked list ที่ระบุโดยอาร์กิวเมนต์ `head_sub` เป็น subset ของ linked list ที่ระบุโดยอาร์กิวเมนต์ `head_super` ถ้าเป็นให้ฟังก์ชันคืนค่า 0 มิฉะนั้นฟังก์ชันจะต้องคืนค่า -1

```
int Linked_List_Is_Subset(S_NODE *head_sub, S_NODE *head_super)
```

นอกจากนี้ในฟังก์ชัน main ให้เขียนโปรแกรมอ่านข้อมูลนำเข้าซึ่งเป็นข้อมูลของนิสิตจำนวน 2 ชุด แต่ละชุดปิดท้ายด้วย “-1 -1 -1” โดยชุดแรกเก็บไว้ที่ linked list ที่ระบุโดย dummy_head_1 ส่วนชุดหลังเก็บไว้ที่ linked list ที่ระบุโดย dummy_head_2

หลังจากนั้นให้เรียกฟังก์ชัน `Linked_List_Is_Subset` เพื่อทำการตรวจสอบว่า linked list ของข้อมูลชุดที่สอง (dummy_head_2) เป็น subset ของ linked list ของข้อมูลชุดแรก (dummy_head_1) หรือไม่ ถ้าเป็นให้แสดง YES มิฉะนั้นให้แสดง NO ก่อนปิดโปรแกรมให้เรียกฟังก์ชัน Free_All_Students เพื่อคืนหน่วยความจำที่ได้จองไว้

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)

`10101 John 60`
`10009 Sarah 50`
`10001 Abby 100`
`10084 Onny 90`
`11101 Jane 15`
`-1 -1 -1`
`10009 Sarah 50`
`10001 Abby 100` ← ข้อมูลชุดที่สองทุกตัวจะเป็นส่วนหนึ่งของข้อมูลแรก
`11101 Jane 15`
`-1 -1 -1`
	YES

`10101 John 60`
`10009 Sarah 50`
`10001 Abby 100`
`10084 Onny 90`
`-1 -1 -1`
`10101 John 60`
`12009 Soe 80` ← ในข้อมูลชุดที่สอง 12009 Soe 80 ไม่เป็นส่วนหนึ่งของของข้อมูลแรก
`-1 -1 -1`
	NO

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct  s_node_tag {
  int ID;
  char name[21];
  int score;
  S_NODE *next;
};

int Add_Student_Ascend(S_NODE *head,int ID,char *name,int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
int Linked_List_Is_Subset(S_NODE *head_sub, S_NODE *head_super);

int main()
{
S_NODE dummy_head_1,dummy_head_2;
int s_id,s_score,retval;
char s_name[21];

      ...=NULL;
      while(1)
      {
            scanf("%d %s %d",&s_id,s_name,&s_score);
            if(s_id==-1)
                  break;
            Add_Student_Ascend(...);
      }
      ...=NULL;
      while(1)
      {
            scanf("%d %s %d",&s_id,s_name,&s_score);
            if(s_id==-1)
                  break;
            Add_Student_Ascend(...);
      }

      retval=Linked_List_Is_Subset(...);
      if(retval==0)
            printf("YES\n");
      else
            printf("NO\n");

      Free_All_Students(...);

      Free_All_Students(...);

      return 0;
}
int Add_Student_Ascend(S_NODE *head,int id,char *name,int score)
{
      ใช้โค้ดข้อก่อนหน้านี้
}
```

```
void Free_All_Students(S_NODE *head)
{
        ใช้โค้ดข้อก่อนหน้านี้
}


int Linked_List_Is_Subset(S_NODE *head_sub, S_NODE *head_super)
{
S_NODE *ptr1,*ptr2;
int found;

        ..........................................          // กำหนดให้ ptr1 ชี้ไปที่โหนดแรกของ head_sub
        while(.........................)          // ไล่วนให้ครบทุกโหนดของ head_sub
        {
                // หาว่ามีโหนดใน head_super ที่มีข้อมูลเหมือนกับ โหนด ptr1

                ..........................................          // กำหนดให้ ptr2 ชี้ไปที่โหนดแรกของ head_super
                found=0;
                while(.........................)          // ไล่วนให้ครบทุกโหนดของ head_super
                {
                        if(.........................)    // ถ้าข้อมูลเหมือนกันแสดงว่าเจอ เปลี่ยนค่า found ออกจากลูปใน
                        {
                                ..........................................
                                ..........................................
                        }
                        ..........................................
                }
                if(found==0)  // ถ้า found เป็น 0 แสดงว่าไม่เจอโหนดที่มีข้อมูลบเหมือนกัน แสดงว่า
                        ..........................................  // head_sub ไม่เป็น subset ของ head_super
                ..........................................
        }
        ..............................// ถ้าหลุดมาถึงต้องนี้แสดงว่าทุกโหนดใน head_sub จะมีโหนดใน head_super ที่มีข้อมูลเหมือนกัน
}
```

3. Problem name: `Linked_List_Update_Score`

จงเขียนโปรแกรม ต่อจากโปรแกรมในข้อ 1 โดยสมมติข้อมูลนำเข้าจะแบ่งเป็นสองส่วน ส่วนแรกนั้นข้อมูล 1 บรรทัดจะประกอบด้วย รหัสและชื่อของนิสิต ถูกป้อนเข้ามาที่ละบรรทัดจนกว่าจะเจอ -1 -1 (ดูตัวอย่างข้างล่าง) หลังจากนั้นข้อมูลในชุดที่สองจะเป็นคะแนนสอบของนิสิตในครั้งต่างๆ โดยนิสิต 1 คนอาจจะมีคะแนนสอบเข้ามามากกว่า 1 ครั้ง โดยข้อมูลส่วนที่สองนั้น 1 บรรทัดจะประกอบด้วย รหัสและคะแนนสอบ ถูกป้อนเข้ามาที่ละบรรทัดจนกว่าจะเจอ -1 -1 (ดูตัวอย่างข้างล่าง)

จงเขียนโปรแกรมเพื่ออ่านข้อมูลทั้งสองชุดเข้ามา โดยการอ่านข้อมูลชุดแรกให้ทำการเพิ่ม node ใน linked list เรียงตามลำดับ ID โดยใช้ฟังก์ชัน `Add_Student_Ascend` ที่เขียนในข้อ 1 โดยให้กำหนดค่าเริ่มต้นของ score ให้เท่ากับ 0 สำหรับทุกๆ node หลังจากนั้นทำการอ่านข้อมูลชุดที่สองแล้วทำการ update ข้อมูล ใน linked list โดยการบวกเพิ่มค่า คะแนนลงในฟิลด์ score ของ node ที่มีค่า ID ตรงต่อรหัสนิสิต โดยให้เขียนและเรียกใช้ฟังก์ชันชื่อ `Update_Score` นิยามดังข้างล่าง ซึ่งจะทำการบวกเพิ่มค่า score ใน node ที่มีค่า ID ตรงกับอาร์กิวเมนต์ s_id ด้วยค่าของ s_score

```
int Update_Score(S_NODE *head, int s_id, int s_score)
```

โดยฟังก์ชันจะคืนค่า -1 ถ้าไม่มี node ใน linked list ที่มีค่า ID ตรงกับค่า s_id และฟังก์ชันจะคืนค่า 0 ถ้าการ update ทำงานได้ถูกต้อง

หลังจากนั้นเมื่ออ่านข้อมูลครบทั้งสองชุดให้ทำการแสดงข้อมูลใน linked list ซึ่งสามารถแสดงดังตัวอย่างข้างล่าง

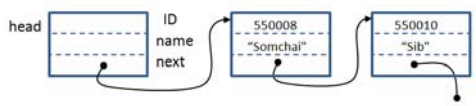
ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
10101 John 10009 Sarah 10001 Abby 10084 Onny 11101 Jane 12009 Soe 12001 Tab 13084 Anny 13184 Peter 13284 Bob -1 -1 11101 15 10009 50 10001 100 10101 60 12001 80 13184 40 10101 60 13284 40 10084 90 11101 15 -1 -1	10001 Abby 100 10009 Sarah 50 10084 Onny 90 10101 John 120 11101 Jane 30 12001 Tab 80 12009 Soe 0 13084 Anny 0 13184 Peter 40 13284 Bob 40

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct  s_node_tag {
  int ID;
  char name[21];
  int score;
  S_NODE *next;
};

int Add_Student_Ascend(S_NODE *head,int ID,char *name,int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
int Update_Score(S_NODE *head,int s_id,int s_score);

int main()
{
S_NODE dummy_head,*ptr;
int s_id,s_score;
char s_name[21];

      dummy_head.next=NULL;
      while(1)
      {
            scanf("%d %s",&s_id,s_name);
            if(s_id==-1)
                  break;
            s_score=0;
            Add_Student_Ascend(    );
      }
      while(1)
      {
            scanf("%d %d",&s_id,&s_score);
            if(s_id==-1)
                  break;
            Update_Score(    );
      }
      Print_All_Students(&dummy_head);
      Free_All_Students(&dummy_head);
      return 0;
}

int Add_Student_Ascend(S_NODE *head,int id,char *name,int score)
void Free_All_Students(S_NODE *head)

      ใช้โค้ดข้อก่อนหน้านี้


int Update_Score(S_NODE *head,int s_id,int s_score)
{
      ดู paper homework
}
```

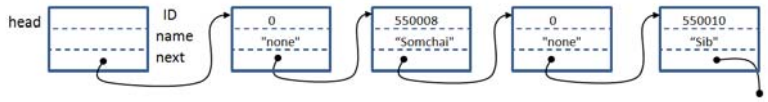
4. Problem name: `Linked_List_Interlace`

จงเขียนโปรแกรม ต่อจากโปรแกรมในข้อ 1 โดยหลังจากอ่านข้อมูลนำเข้าและเก็บลงใน linked list จนเจอบรรทัด -1 -1 -1 แล้ว ให้ทำการแทรก node ใหม่ลงไปข้างหน้า node ทุกๆ node โดยให้ node ที่แทรกลงไปนี้มีข้อมูลเท่ากับข้อมูลที่รับเข้ามา 1 บรรทัดถัดจากข้อมูลชุดแรก

ตัวอย่างเช่น ถ้าข้อมูลใน linked list ปัจจุบันเป็นดังข้างล่างด้านซ้าย หลังจากการแทรก node ใหม่ที่มีข้อมูล ID=0, name="none", score=-1 แล้วจะดัง linked list ดังรูปด้านขวา (หมายเหตุ ในรูปข้างล่างไม่ได้แสดงข้อมูลในฟิลด์ score)



ก่อน



หลังแทรก

กำหนดให้เขียนและเรียกใช้ฟังก์ชัน Interlace ดังนิยามข้างล่าง ซึ่งจะทำการแทรกโหนดใหม่ลงไปข้างหน้าทุกๆโหนด โดยมีข้อมูลเท่ากับอาร์กิวเมนต์ i_id, i_name และ i_score ตามลำดับ

```
void Interlace(S_NODE *head,int i_id,char *i_name,int i_score)
```

เมื่อทำการเรียกฟังก์ชัน `interlace` แล้วให้ทำการเรียกฟังก์ชัน `Print_All_Students` เพื่อทำการแสดงข้อมูลปัจจุบันใน linked list

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
10101 John 60 10009 Sarah 50 10001 Abby 100 10084 Onny 90 11101 Jane 15 -1 -1 -1 0 none -1	0 none -1 10001 Abby 100 0 none -1 10009 Sarah 50 0 none -1 10084 Onny 90 0 none -1 10101 John 60 0 none -1 11101 Jane 15

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct  s_node_tag {
  int ID;
  char name[21];
  int score;
  S_NODE *next;
};

int Add_Student_Ascend(S_NODE *head,int ID,char *name,int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
void Interlace(S_NODE *head,int id,char *name,int score);

int main()
{
S_NODE dummy_head,*ptr;
int i_id,i_score;
char i_name[21];

      dummy_head.next=NULL;
      while(1)
      {
            scanf("%d %s %d",&s_id,s_name,&s_score);
            if(s_id==-1)
                  break;
            Add_Student_Ascend(           );

      }

      scanf("%d %s %d",&i_id,i_name,&i_score);

      Interlace(              );

      Print_All_Students(           );

      Free_All_Students(                  );

      return 0;
}

int Add_Student_Ascend(S_NODE *head,int id,char *name,int score)
void Print_All_Students(S_NODE *head)
void Free_All_Students(S_NODE *head)
      ใช้โค้ดข้อก่อนหน้านี้

void Interlace(S_NODE *head,int i_id,char *i_name,int i_score)
{

      ดู paper homework

}
```

5. Problem name: `Find_LT_Avg_Oldest_Student_2`

จงเขียนโปรแกรมเพื่อค้นหาและแสดงข้อมูลของนักเรียนที่มีอายุมากที่สุดที่น้อยกว่าอายุเฉลี่ยของนักเรียนทั้งหมด

โดยกำหนดให้ข้อมูลนำเข้าในแต่ละบรรทัดคือข้อมูลของนักเรียนแต่ละคนซึ่งประกอบไปด้วย รหัส ชื่อ นามสกุล เพศ อายุ และเกรดเฉลี่ย โดยข้อมูลที่บรรทัดสุดท้ายคือ 0 0 0 0 0 0 จะแสดงจุดสิ้นสุดของข้อมูล (ไม่นับเป็นส่วนกนึ่งของข้อมูล)

โปรแกรมจะแสดงผลลัพธ์เป็นค่าเฉลี่ยของอายุ (ทศนิยม 2 ตำแหน่ง) และข้อมูลของนักเรียนที่ค้นเจอ

<u>นอกจากนี้กำหนดให้เขียนโปรแกรมโดยใช้ตัวแปรแบบ structure ที่เหมาะสมและโครงสร้างข้อมูลแบบ linked list ที่เหมาะสมและ ให้ใช้ reuse โค้ดที่เขียนไปก่อนหน้านี้</u>

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	`530010101 John Jane M 25 3.13`
	`530010009 Sarah Lee F 19 2.98`
	`530010001 Abby Mater M 17 3.10`
	`530010084 Onny Lynch F 21 3.95`
	`530011101 Jane John M 25 2.13`
	`530012009 So Lee-Tung F 20 3.98`
	`530012001 Tab Mccoy M 18 2.10`
	`530013084 Anny Lunch F 22 2.95`
	`0 0 0 0 0 0`

ข้อมูลส่งออก (Output)	`20.88` `530012009 So Lee-Tung F 20 3.98`

<u>หมายเหตุ</u> ในตัวอย่างข้างบนค่าเฉลี่ยของอายุของนักเรียนทั้งหมดมีค่าเท่ากับ 20.88 และนักเรียนที่มีอายุมากที่สุดที่น้อยกว่าอายุเฉลี่ยคือ So Lee-Tung ซึ่งมีอายุเท่ากับ 20

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
typedef struct s_node_tag S_NODE;
struct s_node_tag {
  int ID;
  char fname[21];
  char lname[51];
  char gender;
  int age;
  double gpa;
  S_NODE *next;
};
int Add_Student_Ascend(S_NODE *head,int id,char *fname, char *lname,
                              char gender,int age,double gpa);
void Free_All_Students(S_NODE *head);
int main()
{
S_NODE dummy_head;
int i,N,sum,oldest_i,oldest_age,id,age;
double avg,gpa;
char fname[21],lname[51],gender;
S_NODE *ptr,*oldest_ptr;
        dummy_head.next=NULL;
        while(1)
        {
                scanf("%d %s %s %c %d %lf",&id,fname,lname,&gender,&age,&gpa);
                if(id==0)
                        break;
                Add_Student_Ascend(    );
        }
        // Calculate the mean
        sum=
        N=
        ptr=
        while()
        {
                sum=sum+
                ptr=
                N=
        }
        avg=(double)sum/N;
        // Find the oldest student whose age < avg
        oldest_age=0;
        oldest_ptr=NULL;
        ptr=dummy_head.next;
        while()
        {
                if()
                {
                        oldest_ptr=
                        oldest_age=
                }
                ...;
        }
        printf("%.2f\n",avg);
        printf("%d %s %s %c %d %.2f\n",oldest_ptr->ID, oldest_ptr->fname,
                                       oldest_ptr->lname,oldest_ptr->gender,
                                       oldest_ptr->age,oldest_ptr->gpa);
        Free_All_Students(…);
        return 0;
}
int Add_Student_Ascend(S_NODE *head,int id,char *fname,char *lname,
                                          char gender,int age,double gpa)
{
        Write your code
}
void Free_All_Students(S_NODE *head)
{
        Write your code
}
```