

Topic 3: Linked list (add)

Pointer & structure & linked list add

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

typedef struct reg
{
    int ID;
    int score;
    struct reg *next;
}tReg;
```

Compare to the one in
the last week, this one is
much better and elegant

```
int main (void)
{
    tReg *stu_ptr;
    tReg *head;
    int i;

    stu_ptr = (tReg *) malloc (sizeof(tReg));
    head = stu_ptr; // anchor or head is a must

    stu_ptr->ID = 1;
    stu_ptr->score = 99;
    stu_ptr->next = NULL;
```

```
stu_ptr = add_student(stu_ptr, 20, 40);
stu_ptr = add_student(stu_ptr, 52, 100);
```

```
stu_ptr = head;
while (stu_ptr != NULL)
{
    printf("ID: %d with score: %d \n",
           stu_ptr->ID, stu_ptr->score);
    stu_ptr = stu_ptr -> next;
}
return 0;
}
```

```
tReg * add_student(tReg *p, int ID, int score)
{
    p->next = (tReg *) malloc (sizeof(tReg));
    p->next->ID = ID;
    p->next->score = score;
    p->next->next = NULL;

    return (p->next);
}
```

You should know how to draw nodes'
relationship

Pointer & structure & linked list add

- You should have a structure to manage the linked list
 - Add a new data structure named regHead
 - regHead is for managing the linked list, which contains
 - Two pointers, named front and rear, which directs to reg structure
 - One variable, named count, to record student counts
 - Modify original code to create the management head and other related parts
 - Modify add student function and other related parts
 - Note that the front and rear will direct to the first and last node of the reg structure

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

```
typedef struct reg
{
    int ID;
    int score;
    struct reg *next;
}tReg;
```

```
typedef struct regHead
{
    int count;
    XXX1 *front;
    XXX1 *rear;
}tRegHead;
```

```
void add_student(tRegHead *head_ptr, int ID, int score);
```

```
int main (void)
{
    tReg *stu_ptr;
    tRegHead *head;
    int i;

    head=XXX2;
    head->count = 0;
    head->front = XXX3;
    head->rear = XXX4;
```

**Quiz: Fill below
10 blanks**

```
add_student(head, 20, 40);
add_student(head, 52, 100);
```

```
stu_ptr = XXX5;
for (i =0; i < XXX6; i++)
{
    printf("ID: %d with score: %d \n",
        stu_ptr->ID, stu_ptr->score);
    stu_ptr = stu_ptr -> next;
}
return 0;
}
```

```
void add_student(tRegHead *head_ptr, int ID, int score)
{
    tReg *new_stu_ptr;
    new_stu_ptr = XXX7;
    new_stu_ptr->ID = ID;
    new_stu_ptr->score = score;

    if (head_ptr->count == 0)
    {
        head_ptr->front = XXX8;
    }
    else
    {
        head_ptr->rear->next = XXX9;
    }
    head_ptr->rear = XXX10;
    head_ptr->count ++;
}
```

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

```
typedef struct reg
{
    int ID;
    int score;
    struct reg *next;
}tReg;
```

```
typedef struct regHead
{
    int count;
    tReg *front; // (or struct reg)
    tReg *rear;  // (or struct reg)
}tRegHead;
```

```
void add_student(tRegHead *stu_ptr, int ID, int score);
```

```
int main (void)
{
    tReg *stu_ptr;
    tRegHead *head;
    int i;

    head=(tRegHead *)malloc(sizeof(tRegHead));
    head->count = 0;
    head->front = NULL;
    head->rear = NULL; // (or head->front);
```

Solution

**A much much better
and better style for
linked list!**

```
add_student(head, 20, 40);
add_student(head, 52, 100);
```

```
stu_ptr = head->front;
for (i =0; i < head->count; i++)
{
    printf("ID: %d with score: %d \n",
        stu_ptr->ID, stu_ptr->score);
    stu_ptr = stu_ptr -> next;
}
return 0;
}
```

```
void add_student(tRegHead *head_ptr, int ID, int score)
{
    tReg *new_stu_ptr;
    new_stu_ptr = (tReg *) malloc (sizeof(tReg));
    new_stu_ptr->ID = ID;
    new_stu_ptr->score = score;

    if (head_ptr->count == 0)
    {
        head_ptr->front = new_stu_ptr;
    }
    else
    {
        head_ptr->rear->next = new_stu_ptr;
    }
    head_ptr->rear = new_stu_ptr;
    head_ptr->count ++;
}
```

Week 6-assignment

```
typedef struct num_storage
{
    int number;
    struct num_storage *next;
} tNumStorage;
```

```
typedef struct num_stor_head
{
    int counts;
    struct num_storage *head;
    struct num_storage *tail;
} tNumStorHead;
```

Input numbers arbitrarily!
You can draw and perform insert!

```
Input a number: 4
list->counts: 1
The sorted list: 4
```

```
Input a number: 3
list->counts: 2
The sorted list: 3 4
```

```
Input a number: 1
list->counts: 3
The sorted list: 1 3 4
```

```
Input a number: 2
list->counts: 4
The sorted list: 1 2 3 4
```

```
Input a number: 4
list->counts: 5
The sorted list: 1 2 3 4 4
```

```
Input a number: 9
list->counts: 6
The sorted list: 1 2 3 4 4 9
```

```
Input a number: 1
list->counts: 7
The sorted list: 1 1 2 3 4 4 9
```

```
Input a number: 0
list->counts: 8
The sorted list: 0 1 1 2 3 4 4 9
```

```
Input a number: -1
```

- You should implement these four functions

```
void initial_list(tNumStorHead *list);  
void get_input(tNumStorHead *list);  
void print_list(tNumStorHead *list);  
void sort_list(tNumStorHead *list, int input);
```

- In the get_input function
 - Perform scanf
 - Call sort_list function
 - Stop when detect a "-1"

```
int main (void)  
{  
    int i;  
    tNumStorHead *list;  
    list = (tNumStorHead *) malloc (sizeof(tNumStorHead));  
    initial_list(list);  
    get_input(list);  
}  
  
void initial_list(tNumStorHead *list)  
{  
    list->counts = 0;  
    list->head = NULL;  
    list->tail = NULL;  
}  
  
void print_list(tNumStorHead *list)  
{  
    tNumStorage *node_ptr = list->head;  
  
    printf(" The sorted list: ");  
    while (node_ptr != NULL)  
    {  
        printf("%d ", node_ptr->number);  
        node_ptr = node_ptr->next;  
    }  
    printf("\n\n");  
}
```