```c
#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 *head_ptr, int ID, int score);
void rotate_student(tRegHead *head_ptr);
void print_student(tRegHead *head_ptr);
```

```c
int main (void)
{

    tRegHead *head;
    int i;

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

    add_student(head, 20, 40);
    add_student(head, 52, 100);
    add_student(head, 32, 50);

    print_student(head);

    rotate_student(head);
    print_student(head);


    return 0;
}
```

After calling rotate_student

ID: 32 with score: 50
ID: 20 with score: 40
ID: 52 with score: 100

→

ID: 32 with score: 50
ID: 20 with score: 40
ID: 52 with score: 100

Implement rotate_student function

ID: 32 with score: 50
ID: 20 with score: 40
ID: 52 with score: 100

Change to

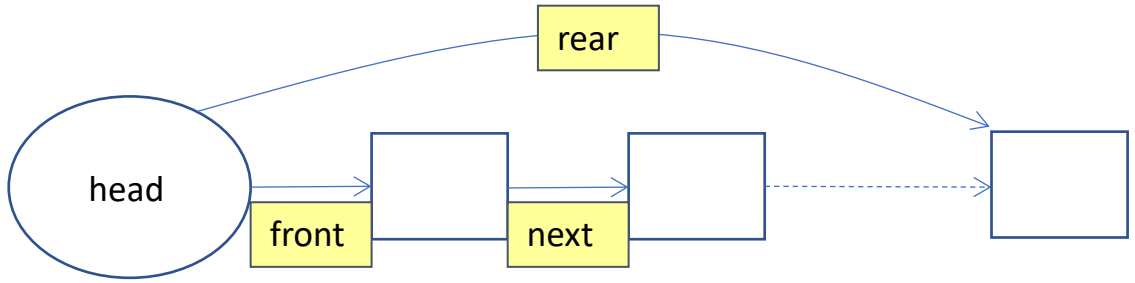
ID: 32 with score: 50
ID: 20 with score: 40
ID: 52 with score: 100

```
void rotate_student(tRegHead *head_ptr)
{
    int i;
    tReg *prev_rear = head_ptr->front;
    tReg *target = head_ptr->rear;

    if (head_ptr->count < 2)
    {
        return;
    }

    for (i=0; i < head_ptr->count - 2; i++)
    {
        prev_rear = _____1_____ ;
    }

    _____________2_____________________
    _____________3_____________________
    _____________4_____________________
    _____________5_____________________
}
```

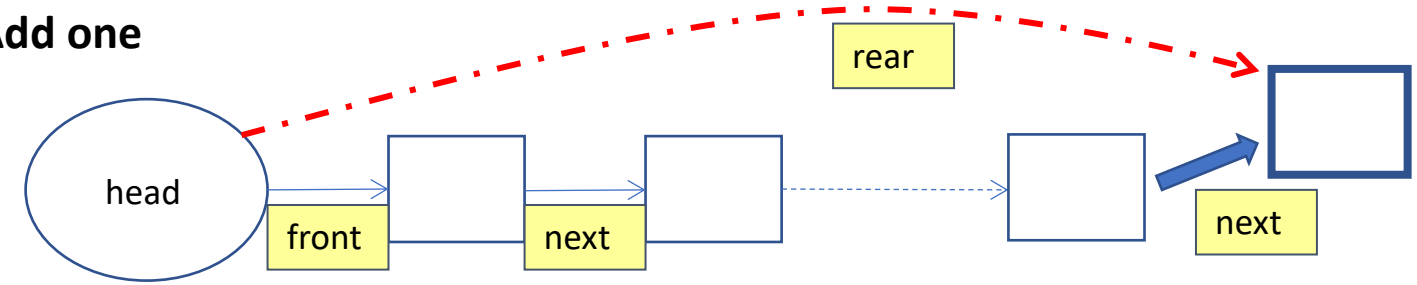
```
void rotate_student(tRegHead *head_ptr)
{
    int i;
    tReg *prev_rear = head_ptr->front;
    tReg *target = head_ptr->rear;

    if (head_ptr->count < 2)
    {
        return;
    }

    for (i=0; i < head_ptr->count - 2; i++)
    {
        prev_rear = prev_rear->next;
    }

    target->next = head_ptr->front;
    head_ptr->front = target;
    prev_rear->next = NULL;
    head_ptr->rear=prev_rear;
}
```

Topic 4: Linked list (remove)

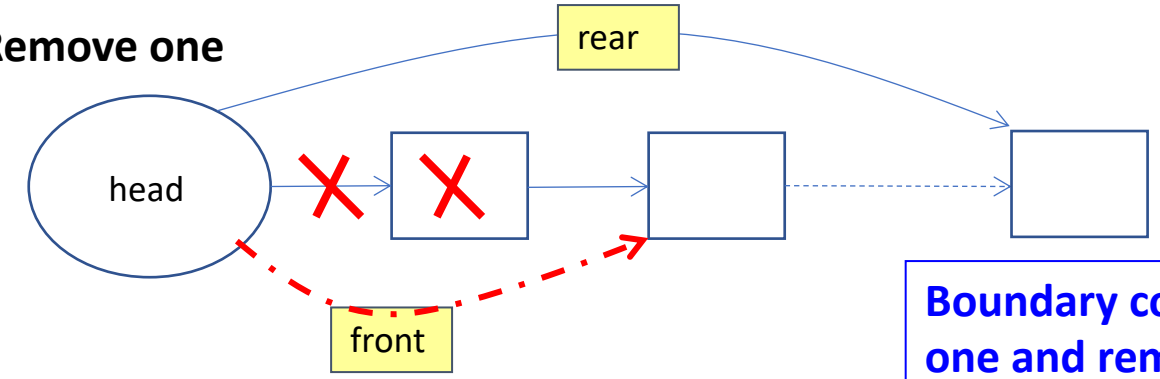
Linked list (Add to last and remove from first → FIFO)



Add one



Remove one



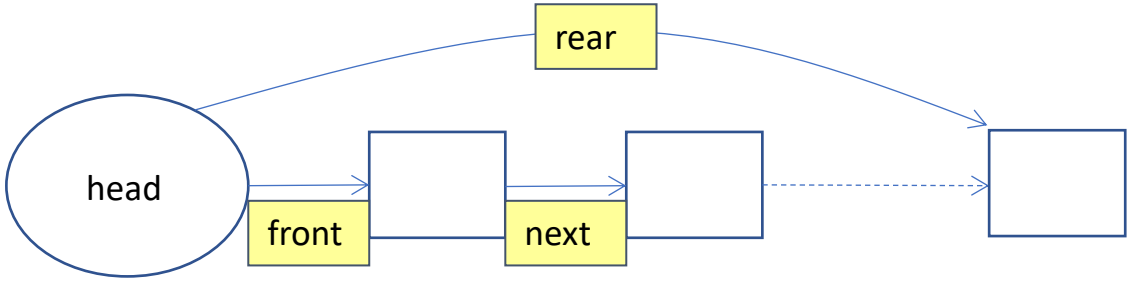
Boundary condition (Add the first one and remove the last one)

Pointer & structure & linked list remove

- Remove the first one

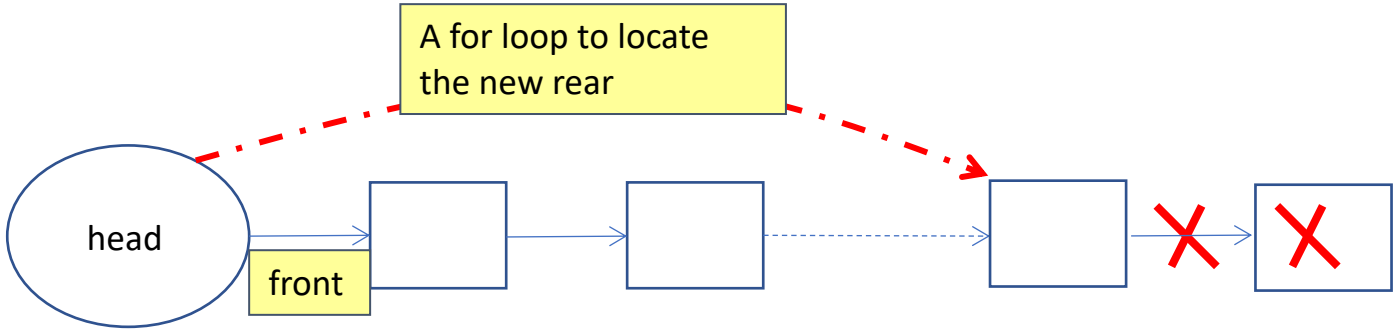
```
void RemStudent (tRegHead *p)
{
    tReg *stu_ptr;

    stu_ptr = p->front;
    p->front = stu_ptr->next;
    p->count --;

    printf ("Remove student ID: %d with score: %d \n",
                stu_ptr->ID, stu_ptr->score);

    free (stu_ptr);
}
```

Linked list (Add to last and remove from last → Stack)



Add one (same as FIFO)

Remove the last one



W7-assignment

- Based on your program last week!
- Implement a delete_last function

```
ryanpan@RyanPanPC /Volumes/MyWorks/D_Data/teaching/1
Input a number (-1 to exit, -2 to delete last): 5
  list->counts: 1
  The sorted list: 5

Input a number (-1 to exit, -2 to delete last): 10
  list->counts: 2
  The sorted list: 5 10

Input a number (-1 to exit, -2 to delete last): 20
  list->counts: 3
  The sorted list: 5 10 20

Input a number (-1 to exit, -2 to delete last): 15
  list->counts: 4
  The sorted list: 5 10 15 20

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 3
  The sorted list: 5 10 15

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 2
  The sorted list: 5 10

Input a number (-1 to exit, -2 to delete last): 15
  list->counts: 3
  The sorted list: 5 10 15

Input a number (-1 to exit, -2 to delete last): 13
  list->counts: 4
  The sorted list: 5 10 13 15

Input a number (-1 to exit, -2 to delete last): 20
  list->counts: 5
  The sorted list: 5 10 13 15 20
```

```
Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 4
  The sorted list: 5 10 13 15

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 3
  The sorted list: 5 10 13

Input a number (-1 to exit, -2 to delete last): 1
  list->counts: 4
  The sorted list: 1 5 10 13

Input a number (-1 to exit, -2 to delete last): 0
  list->counts: 5
  The sorted list: 0 1 5 10 13

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 4
  The sorted list: 0 1 5 10

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 3
  The sorted list: 0 1 5

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 2
  The sorted list: 0 1

Input a number (-1 to exit, -2 to delete last): -4
  list->counts: 3
  The sorted list: -4 0 1

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 2
  The sorted list: -4 0
```

```
Input a number (-1 to exit, -2 to delete last): 1
  list->counts: 3
  The sorted list: -4 0 1

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 2
  The sorted list: -4 0

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 1
  The sorted list: -4

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 0
  The sorted list:

Input a number (-1 to exit, -2 to delete last): 9
  list->counts: 1
  The sorted list: 9

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 0
  The sorted list:

Input a number (-1 to exit, -2 to delete last): 4
  list->counts: 1
  The sorted list: 4

Input a number (-1 to exit, -2 to delete last): -2
  list->counts: 0
  The sorted list:

Input a number (-1 to exit, -2 to delete last): -2
    There is nothing to delete

Input a number (-1 to exit, -2 to delete last): -1
```

- Add a new function delete_last

```
void delete_last(tNumStorHead *list);
```

- No while loop or for loop in the delete_last function is allowed !!!
  - Think necessary modifications by yourself

```
void get_input(tNumStorHead *list)
{
    int input = 0, result;

    while (input != -1)
    {
        printf("Input a number (-1 to exit, -2 to delete last): ");
        scanf("%d", &input);
        if (input == -2)
        {
            delete_last(list);
        }
        else if (input != -1)
        {
            sort_list (list, input);
        }
    }
}
```