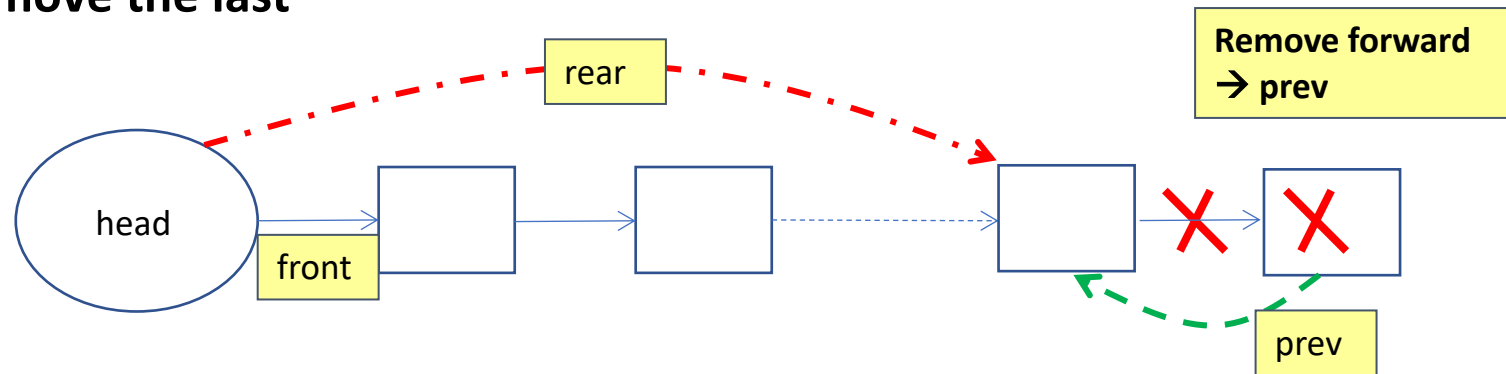


# Topic 4: Linked list (remove)

# Linked list (remove the last → stack)

Remove the last



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

Boundary condition (Add the first or remove the last)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct reg
5  {
6      int ID;
7      int score;
8      struct reg *next;
9      struct reg *prev; ←
10 }tReg;
11
12 typedef struct regHead
13 {
14     int count;
15     tReg *front; // (or struct reg)
16     tReg *rear;  // (or struct reg)
17 }tRegHead;
18
19 void add_student(tRegHead *head_ptr, int ID, int score);
20 void rem_student(tRegHead *head_ptr);
```

```

22 int main (void)
23 {
24     tReg *stu_ptr;
25     tRegHead *head_ptr;
26     int i;
27
28     head_ptr=(tRegHead *)malloc(sizeof(tRegHead));
29     head_ptr->count = 0;
30     head_ptr->front = NULL;
31     head_ptr->rear = NULL; // (or head_ptr->front);
32
33     add_student(head_ptr, 20, 40);
34     add_student(head_ptr, 52, 100);
35
36     stu_ptr = head_ptr->front;
37     for (i =0; i < head_ptr->count; i++)
38     {
39         printf("ID: %d with score: %d \n",
40             stu_ptr->ID, stu_ptr->score);
41         stu_ptr = stu_ptr -> next;
42     }
43
44     while (head_ptr->count)
45     {
46         rem_student(head_ptr);
47     }
48 }

```

```

50 void add_student(tRegHead *head_ptr, int ID, int score)
51 {
52     tReg *new_stu_ptr;
53     new_stu_ptr = (tReg *) malloc (sizeof(tReg));
54     new_stu_ptr->ID = ID;
55     new_stu_ptr->score = score;
56     new_stu_ptr->next = NULL;
57     new_stu_ptr->prev = NULL;
58
59     if (head_ptr->count == 0)
60     {
61         head_ptr->front = new_stu_ptr;
62     }
63     else
64     {
65         head_ptr->rear->next = new_stu_ptr;
66         new_stu_ptr->prev = head_ptr->rear;
67     }
68     head_ptr->rear = new_stu_ptr;
69     head_ptr->count ++;
70 }

```

# Remove the last

```
3
4 typedef struct reg
5 {
6     int ID;
7     int score;
8     struct reg *next;
9     struct reg *prev;
10 }tReg;
11
12 typedef struct regHead
13 {
14     int count;
15     tReg *front; // (or struct reg)
16     tReg *rear; // (or struct reg)
17 }tRegHead;
18
19 void add_student(tRegHead *head_ptr, int ID, int score);
20 void rem_student(tRegHead *head_ptr);
```

```
22 int main (void)
23 {
24     tReg *stu_ptr;
25     tRegHead *head_ptr;
26     int i;
27
28     head_ptr=(tRegHead *)malloc(sizeof(tRegHead));
29     head_ptr->count = 0;
30     head_ptr->front = NULL;
31     head_ptr->rear = NULL; // (or head_ptr->front);
32
33     add_student(head_ptr, 20, 40);
34     add_student(head_ptr, 52, 100);
35
36     stu_ptr = head_ptr->front;
37     for (i =0; i < head_ptr->count; i++)
38     {
39         printf("ID: %d with score: %d \n",
40             stu_ptr->ID, stu_ptr->score);
41         stu_ptr = stu_ptr -> next;
42     }
43
44     while (head_ptr->count)
45     {
46         rem_student(head_ptr);
47     }
```

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

    stu_ptr = XX1;

    if (head_ptr->count > 1)
    {
        head_ptr->rear = XX2;
        head_ptr->rear->next = XX3;
    }
    else
    {
        head_ptr->rear = XX4;
        head_ptr->front = XX5;
    }
    head_ptr->count --;

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

    free (stu_ptr);
}
```

Quiz

# Remove the last

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

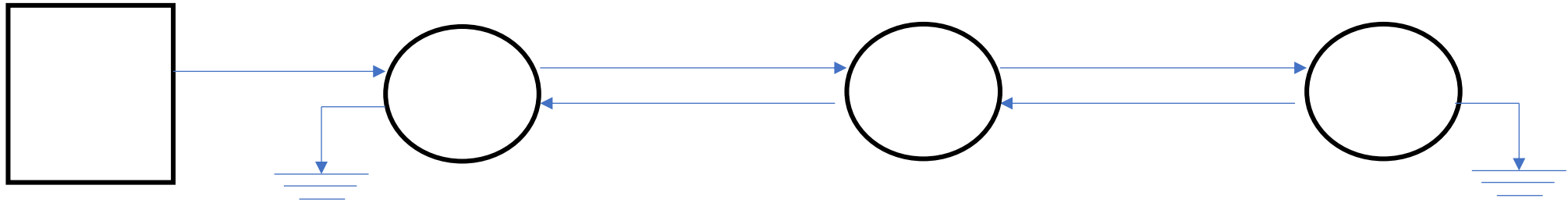
    stu_ptr = head_ptr->rear;

    if (head_ptr->count > 1)
    {
        head_ptr->rear = stu_ptr->prev;
        head_ptr->rear->next = NULL;
    }
    else
    {
        head_ptr->rear = NULL;
        head_ptr->front = NULL;
    }
    head_ptr->count --;

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

    free (stu_ptr);
}
```

# Bidirectional linked list: add remove randomly



# W7-assignment

## Two stages

- First stage:

- Use your own program last week
- Modify the print\_list function as the right one → your printList function should be the same as mine in this assignment
- Try first during the class !

```
list->counts: 14
The sorted list: 1 1 2 2 4 4 5 5 8 8 8 8 8 9
The sorted list: 9 8 8 8 8 8 5 5 4 4 2 2 1 1
0
list->counts: 15
The sorted list: 0 1 1 2 2 4 4 5 5 8 8 8 8 8 9
The sorted list: 9 8 8 8 8 8 5 5 4 4 2 2 1 1 0
5
list->counts: 16
The sorted list: 0 1 1 2 2 4 4 5 5 5 8 8 8 8 8 9
The sorted list: 9 8 8 8 8 8 5 5 5 4 4 2 2 1 1 0
```

```
void print_list(tNumStorHead *list)
{
    tNumStorage *node_ptr;

    printf("\n");
    printf("counts: %d \n", list->counts);
    node_ptr = list->head;
    printf (" From head --> ");
    while (node_ptr != NULL)
    {
        printf("%d ", node_ptr->number);
        node_ptr = node_ptr->next;
    }
    node_ptr = list->tail;
    printf ("\n From tail --> ");
    while (node_ptr != NULL)
    {
        printf("%d ", node_ptr->number);
        node_ptr = node_ptr->prev;
    }
    printf("\n");
}
```



# Second stage

- [Upload this one only](#)
- Add or remove randomly →→
- Before your delivery, test your program following my procedure !!

```
1 1. Add a number or 2. Delete a number: 1
2 Add a number: 1
3
4 count: 1
5 From head --> 1
6 From tail --> 1
7
8 1. Add a number or 2. Delete a number: 1
9 Add a number: 2
10 Specify a target location: 1
11 1. Before or 2. After the location 1: 2
12
13 count: 2
14 From head --> 1 2
15 From tail --> 2 1
16
17 1. Add a number or 2. Delete a number: 3
18 No such choose !
19 1. Add a number or 2. Delete a number: 1
20 Add a number: 3
21 Specify a target location: 1
22 1. Before or 2. After the location 1: 1
23
24 count: 3
25 From head --> 3 1 2
26 From tail --> 2 1 3
27
28 1. Add a number or 2. Delete a number: 1
29 Add a number: 4
30 Specify a target location: 3
31 1. Before or 2. After the location 3: 2
32
33 count: 4
34 From head --> 3 1 2 4
35 From tail --> 4 2 1 3
```

```
1 1. Add a number or 2. Delete a number: 1
2 Add a number: 5
3 Specify a target location: 2
4 1. Before or 2. After the location 2: 1
5
6 count: 5
7 From head --> 3 5 1 2 4
8 From tail --> 4 2 1 5 3
9
10 1. Add a number or 2. Delete a number: 2
11 Specify a target location: 3
12
13 count: 4
14 From head --> 3 5 2 4
15 From tail --> 4 2 5 3
16
17 1. Add a number or 2. Delete a number: 2
18 Specify a target location: 4
19
20 count: 3
21 From head --> 3 5 2
22 From tail --> 2 5 3
23
24 1. Add a number or 2. Delete a number: 1
25 Add a number: 6
26 Specify a target location: 3
27 1. Before or 2. After the location 3: 2
28
29 count: 4
30 From head --> 3 5 2 6
31 From tail --> 6 2 5 3
32
33 1. Add a number or 2. Delete a number: 2
34 Specify a target location: 4
35
36 count: 3
37 From head --> 3 5 2
38 From tail --> 2 5 3
```

```

76 1. Add a number or 2. Delete a number: 1
77 Add a number: 6
78 Specify a target location: 3
79 1. Before or 2. After the location 3: 1
80
81 count: 4
82 From head --> 3 5 6 2
83 From tail --> 2 6 5 3
84
85 1. Add a number or 2. Delete a number: 2
86 Specify a target location: 1
87
88 count: 3
89 From head --> 5 6 2
90 From tail --> 2 6 5
91
92 1. Add a number or 2. Delete a number: 2
93 Specify a target location: 2
94
95 count: 2
96 From head --> 5 2
97 From tail --> 2 5
98
99 1. Add a number or 2. Delete a number: 1
100 Add a number: 6
101 Specify a target location: 2
102 1. Before or 2. After the location 2: 2
103
104 count: 3
105 From head --> 5 2 6
106 From tail --> 6 2 5

```

```

108 1. Add a number or 2. Delete a number: 2
109 Specify a target location: 2
110
111 count: 2
112 From head --> 5 6
113 From tail --> 6 5
114
115 1. Add a number or 2. Delete a number: 1
116 Add a number: 9
117 Specify a target location: 2
118 1. Before or 2. After the location 2: 1
119
120 count: 3
121 From head --> 5 9 6
122 From tail --> 6 9 5
123
124 1. Add a number or 2. Delete a number: 2
125 Specify a target location: 1
126
127 count: 2
128 From head --> 9 6
129 From tail --> 6 9
130
131 1. Add a number or 2. Delete a number: 2
132 Specify a target location: 1
133
134 count: 1
135 From head --> 6
136 From tail --> 6
137
138 1. Add a number or 2. Delete a number: 2
139 Specify a target location: 1
140
141 count: 0
142 From head -->
143 From tail -->

```

```

145 1. Add a number or 2. Delete a number: 1
146 Add a number: 1
147
148 count: 1
149 From head --> 1
150 From tail --> 1
151
152 1. Add a number or 2. Delete a number: 1
153 Add a number: 2
154 Specify a target location: 1
155 1. Before or 2. After the location 1: 2
156
157 count: 2
158 From head --> 1 2
159 From tail --> 2 1
160
161 1. Add a number or 2. Delete a number: 1
162 Add a number: 3
163 Specify a target location: 1
164 1. Before or 2. After the location 1: 1
165
166 count: 3
167 From head --> 3 1 2
168 From tail --> 2 1 3

```