4190.103A-001: Programming Practice Lab. 12

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2023

Lab. 11 실습 풀이

실습 1

영수는 이번 달에 N번의 카드 결제를 진행했다. 따라서 N개의 결제 대금이 입력으로 들어올 것이다. 영수는 총 금액을 계산하기 위해 PAYMENT 구조체를 이용하기로 하였다. PAYMENT 구조체 안의 _10s, _100s, _1000s, _10000s 변수는 총 지불해야 할 금액의 10원, 100원, 1000원, 10000원짜리 개수를 나타낸다. _10s, _100s, _1000s 변수의 경우(10원, 100원, 1000원 짜리의 경우) 0~9 사이의 값을 가지고, _10000s는 0 이상의 값을 가진다고 할 때 PAYMENT 구조체를 이용하여 총 지불해야할 금액의 10원, 100원, 1000원, 10000원짜리 개수를 구하시오. (모든 금액은 10원의 배수이다.)

```
void update_payment(PAYMENT * pay, int input) {
    // 1.3 _10s, _100s, _1000s, _10000s 금액을 
        // 1.3.1 input 값을 10000원, 1000원, 10
        // 1.3.2 10원부터 10000원까지 *pay 구조
        //       이 과정에서 10원, 100원, 1000원
        //       예를 들어, 10원짜리가 10개인 경
    // ### WRITE YOUR CODE ###
    int _10000s = input / 10000;
    input = input % 10000;
    int _1000s = input / 1000;
    input = input % 1000;
    int _100s = input / 100;
    input = input % 100;
    int _10s = input / 10;

    pay->_10s += _10s;
    if (pay->_10s >= 10) {
        pay->_10s -= 10;
        pay->_100s++;
    }

    pay->_100s += _100s;
    if (pay->_100s >= 10) {
        pay->_100s -= 10;
        pay->_1000s++;
    }

    pay->_1000s += _1000s;
    if (pay->_1000s >= 10) {
        pay->_1000s -= 10;
        pay->_10000s++;
    }

    pay->_10000s += _10000s;
}
```

실습 2

총 N명의 학생이 있다. 입력의 첫 줄로 학생의 수 N이 주어지고, 두 번째 줄에는 N명의 학생들의 학번이 주어진다. 세 번째 줄에는 N명의 학생들의 시험 성적이 주어진다. STUDENT 구조체 배열을 선언하여 학생의 정보를 입력받고, 성적이 낮은 순으로 정렬 한 후, 학생의 학번을 출력하자. 만약 성적이 같은 경우, 학번이 낮은 순으로 정렬하면 된다.

```c
int main(void) {
    int N;
    scanf("%d", &N);
    // ### WRITE YOUR CODE HERE
    STUDENT* list = (STUDENT*)malloc(N * sizeof(STUDENT));

    for(int i = 0; i < N; i++)
    {
        scanf("%d", &list[i].student_id);
    }

    for(int i = 0; i < N; i++)
    {
        scanf("%d", &list[i].score);
    }

    sort(list, N);

    for(int i = 0; i < N; i++)
    {
        printf("%d ", list[i].student_id);
    }

    return 0;
}
```

```c
void sort(STUDENT * students, int N) {
    // ### WRITE YOUR CODE HERE
    for (int i = 0; i < N - 1; i++)
        for (int j = N - 1; j > i; j--)
            if (students[j-1].score > students[j].score)
            {
                swap(&students[j-1], &students[j]);
            }
            else if (students[j-1].score == students[j].score)
            {
                if (students[j-1].student_id > students[j].student_id)
                    swap(&students[j-1], &students[j]);
            }
}
```

실습 3

전자 도서관 프로그램을 만들어보자. 전자 도서관 프로그램을 만들기 위해서는 2개의 구조체를 선언해야 한다. 첫 째로 Book 구조체는 책 제목(title), 저자(author), 출판년도(year) 항목을 가진다. title과 author의 최대 길이는 99이다. 둘째로 Library 구조체는 Book 변수모음인 Book 배열과 총 책의 수인 int형 변수를 하나 가진다.

전자 도서관 프로그램은 아래와 같은 4가지 작업을 제공한다.

A : 책의 title, author, year를 정보로 받고, 앞서 명시한 자료 구조를 사용하여 책을 전자 도서관에 등록할 것.
D : 마지막에 등록된 책의 정보를 삭제할 것. 만약 책이 없다면 "Error\n"를 출력할 것.
S : 마지막에 등록된 책의 정보를 출력할 것. 만약 책이 없는 경우 "None\n"을 출력할 것.
Q : 프로그램을 종료시킬 것.

```c
int main(void) {
    Library lib;
    lib.count = -1;
    char option;
    int cond = 1;
    Book b;

    while (cond) {
        scanf(" %c", &option);
        switch(option) {
            case 'A' :
                scanf("%s %s %d", b.title, b.author, &b.year);
                lib.books[++lib.count] = b;
                break;
            case 'D' :
                if (lib.count < 0)
                    printf("Error\n");
                else
                    lib.count--;
                break;
            case 'S' :
                if (lib.count >= 0)
                    printf("%s %s %d\n", lib.books[lib.count].title,
                            lib.books[lib.count].author, lib.books[lib.count].year);
                else
                    printf("None\n");
                break;
            case 'Q' :
                cond = 0;
                break;
        }
    }
    return 0;
}
```

실습 4

학교에는 적재할 수 있는 무게가 각각 다른 사물함들이 있다. 학생들의 책가방의 무게는 서로 다르다. 만약 적재할 수 있는 무게를 초과하여 사물함에 책가방을 넣으면 사물함이 부서질 수 있다고 한다. 최대 책가방의 무게는 POSSIBLE_MAX_LOAD로 10000이다. 총 N개의 사물함과 M명의 학생들이 있으며, N, M 모두 100 이하라고 가정하자.

첫 줄에는 입력으로 N과 M이 입력된다. 둘 째 줄에는 입력으로 각 사물함마다 적재할 수 있는 최대 무게가 주어진다. 마지막으로 셋 째 줄에는 입력으로 학생들의 책가방의 무게가 순서대로 주어진다. 학생 번호 (student_id)가 작은 순서대로 먼저 사물함을 고를 기회가 주어진다. 학생들은 사물함이 부서지지 않는 선에서 적제 가능 무게와 책가방의 무게 차가 가장 작은 사물함을 선택해야 한다고 한다. M명의 학생들의 cabinet_id를 차례대로 출력하시오.

```
void find_best_cabinet(CABINET * cabinets, int N, STUDENT * student) {
    int smallest_idx = 0;
    int smallest_value = POSSIBLE_MAX_LOAD;
    for (int i = 0; i < N; i++) {
        if(cabinets[i].taken == 0)
            if(cabinets[i].max_load >= student->backpack_weight)
                if(cabinets[i].max_load - student->backpack_weight < smallest_value)
                {
                    smallest_value = cabinets[i].max_load - student->backpack_weight;
                    smallest_idx = i;
                }
    }

    student->cb = &(cabinets[smallest_idx]);
    cabinets[smallest_idx].taken = 1;
}
```

Lab. 11 과제 풀이

과제 1

Q. 야구 선수

야구 에이전트 H 소속 매니저는, 야구 선수들 명단을 특정 기준으로 정렬된 결과에 따라 보고 싶어 한다. 야구 선수 N명의 이름, 키, 타율, 홈런수를 입력받은 후 입력된 명령어(`1,2,3,4`)에 따라 정렬된 결과를 출력해보자.

입력

- 첫째줄에 N이 주어진다.
- 다음 N개의 줄에 이름, 키, 타율, 홈런수가 공백으로 구분되어 입력된다.
- 이후 문자 'q'가 입력될 때까지 명령어(1 or 2 or 3 or 4)를 입력받고 알맞게 출력한다.

출력

- 명령어 1이 입력되면, 이름을 기준으로 사전순으로 정렬된 결과를 출력한다. ('A'~'Z'순서,첫글자가 같으면 그 다음 글자 비교)
- 명령어 2가 입력되면, 키를 기준으로 내림차순 정렬된 결과를 출력한다.
- 명령어 3이 입력되면, 타율을 기준으로 내림차순 정렬된 결과를 출력한다.
- 명령어 4가 입력되면, 홈런수를 기준으로 내림차순 정렬된 결과를 출력한다.
  (모든 출력은 **선수의 이름**만 출력한다.)

```c
int main(){
    int N,i;
    char command = '\0';
    Player *p_list;

    scanf("%d",&N);
    p_list = (Player*)malloc(sizeof(Player)*N);

    for(i=0;i<N;i++){
        /*
            Step 1 ) N만큼 선수에 대한 정보를 입력받아 p_list에 저장.
        */
        p_list[i] = (Player)malloc(sizeof(player)); // Player 는 player* 이므로, p_list[i]는 포인터.
        scanf("%s %d %f %d",p_list[i]->name,&(p_list[i]->height),&(p_list[i]->bat_avg),&(p_list[i]->homerun_cnt));
    }
    /*
        Step 2 ) 'q'가 입력될 때까지 명령어 입력받고 양식에 맞게 출력.
    */
    while(1){
        scanf(" %c",&command);
        if(command == 'q')
            break;
        switch(command){
            case '1' : sortByName(N,p_list);break;
            case '2' : sortByHeight(N,p_list);break;
            case '3' : sortByBat(N,p_list);break;
            case '4' : sortByHomerun(N,p_list);break;
            default : break;
        }
        printPlayers(N,p_list);
    }
}
```

과제 1 - 2

```c
void sortByName(int num, Player *list){
    for(int i=0;i<num-1;i++){
        for(int j=i+1;j<num;j++){
            if(strcmp(list[i]->name,list[j]->name) > 0){
                Player temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
        }
    }
}
void sortByHeight(int num, Player *list){
    for(int i=0;i<num-1;i++){
        for(int j=i+1;j<num;j++){
            if(list[i]->height < list[j]->height){
                Player temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
        }
    }
}
```

```c
void sortByBat(int num, Player *list){
    for(int i=0;i<num-1;i++){
        for(int j=i+1;j<num;j++){
            if(list[i]->bat_avg < list[j]->bat_avg){
                Player temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
        }
    }
}
void sortByHomerun(int num, Player *list){
   for(int i=0;i<num-1;i++){
        for(int j=i+1;j<num;j++){
            if(list[i]->homerun_cnt < list[j]->homerun_cnt){
                Player temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
        }
    }
}
```

과제 2

Q. 포커 게임

포커를 플레이하며 "스트레이트(Straight)" 가 나올 확률을 구해봅시다.

6명의 플레이어가 포커를 플레이 할 때,
스트레이트가 나올 확률을 계산하는 프로그램을 작성 해 봅시다.

포커에서 스트레이트란,
받은 5개의 패가 모두 연속한 숫자일 때를
스트레이트라고 합니다.
그림의 모양이나 색깔은 상관없으며, 숫자만 고려하여
연속인지 판단합니다.
이 때, 'A' 카드는 'K' 뒤에 올 수도 있고,
'2' 앞에 올 수도 있습니다.
즉,
A, 2, 3, 4, 5 도 스트레이트고,
10, J, Q, K, A 도 스트레이트 입니다.
그러나, "K, A, 2, 3, 4" 와 같이 'A'가 중간에 오는
상황은 스트레이트가 아닙니다.
(- 문제에서 A는 1로 표현하고, J,Q,K 는 각각 11, 12, 13으로 표현합니다.)

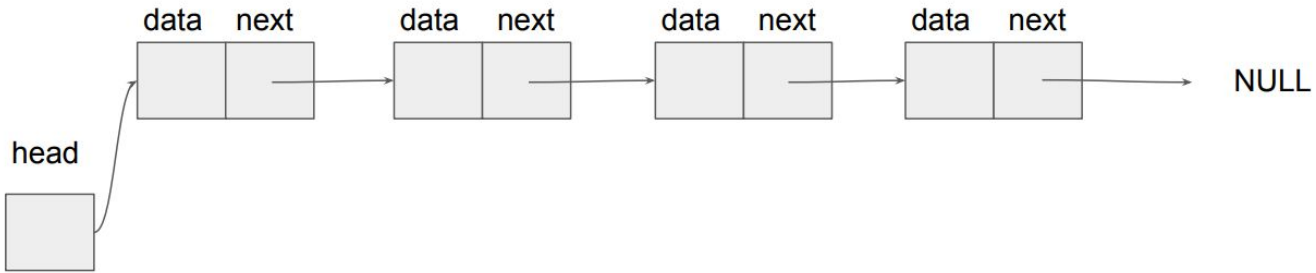
```
int is_straight(card* hand)
{
    /* Write your code here */
    sort(hand);

    if(hand[0].pips == 1 && hand[1].pips == 10 && hand[2].pips == 11
        && hand[3].pips == 12 && hand[4].pips == 13)
        return 1;

    for(int i = 1; i < NHANDS; i++)
        if(hand[i].pips != hand[i - 1].pips + 1)
            return 0;

    return 1;
}
```

```
void sort(card *a)
{
    for (int i = 0; i < 5 - 1; i++)
        for (int j = 5 - 1; j > i; j--)
            if (a[j-1].pips > a[j].pips)
            { /* swap a[j-1], a[j] */
                card tmp = a[j-1];
                a[j-1] = a[j];
                a[j] = tmp;
            }
}
```

실습

실습 1 - Singly linked list

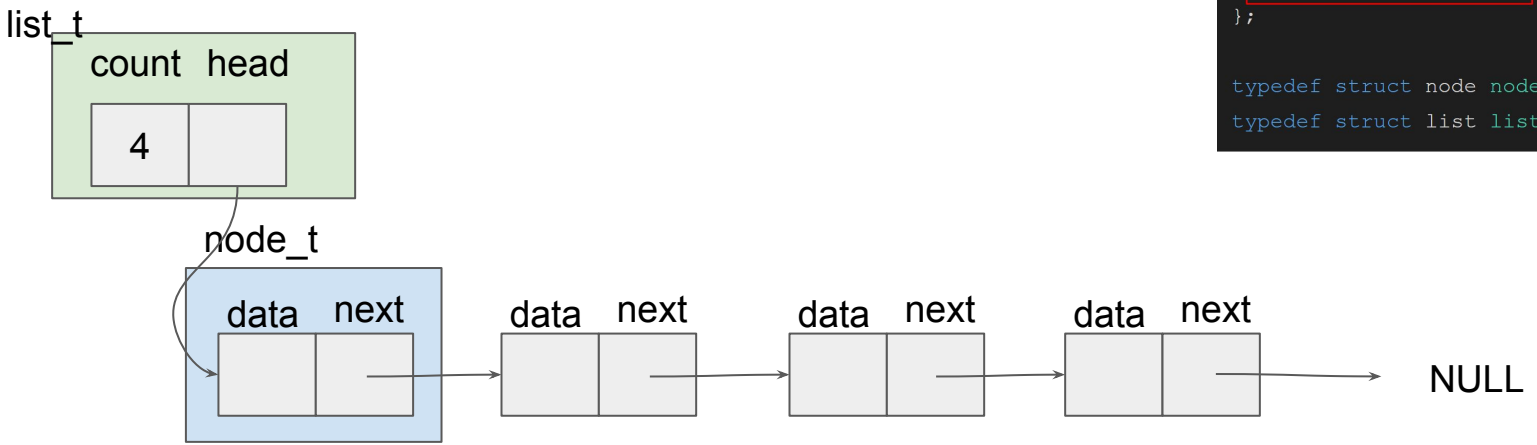
- Singly linked list
  - Element간의 연결을 통해 리스트를 구현한 자료구조
  - 노드들이 한쪽 방향으로 만 연결이 되어있음
- Interface
  - insert_node_first(), insert_node_last(), insert_node(), delete_node(), print_list(), etc…



실습 1 - Singly linked list

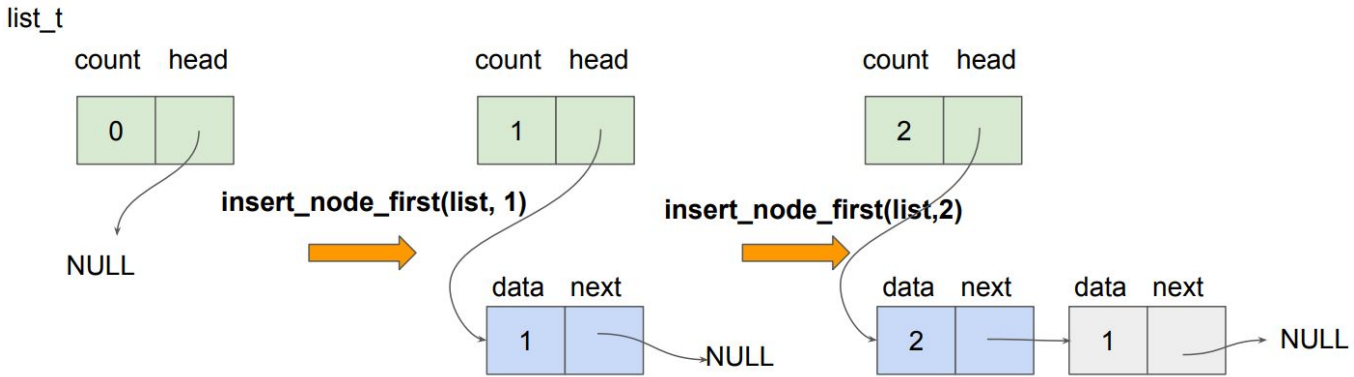
- Singly linked list를 위한 structure를 정의해봅시다.
- 각 node는 다음 node를 가리키는 포인터가 필요합니다.
- List의 첫번째 노드를 가리키는 포인터 또한 필요합니다.

```c
struct node {
  int data;
  struct node * next;
};

struct list {
  int count;
  struct node * head;
};

typedef struct node node_t;
typedef struct list list_t;
```



실습 1 - insert_node_first

- List의 가장 앞쪽에 node를 삽입하는 함수를 함께 구현해 봅시다
  - int insert_node_first(list_t *list, int data)

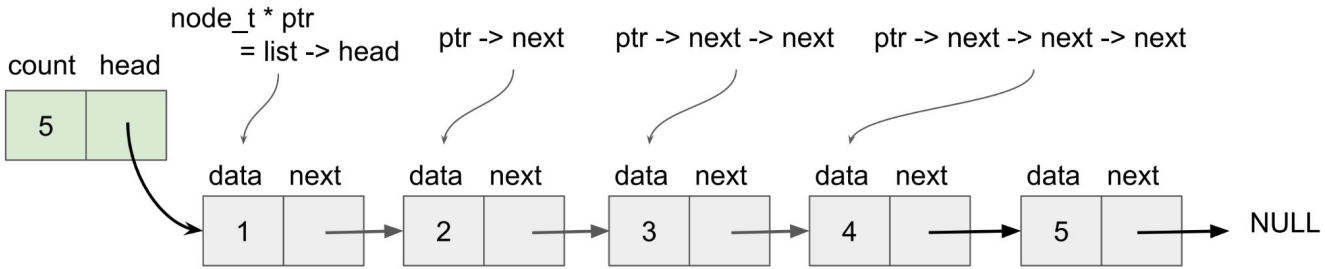


실습 1 - insert_node_first

- List의 가장 앞쪽에 node를 넣으려면?
  - 1. 노드하나를 생성함
  - 2. 생성한 노드의 next를 head가 가리키고 있는 노드로 변경
  - 3. list의 head를 현재 생성한 노드를 가리키게 변경

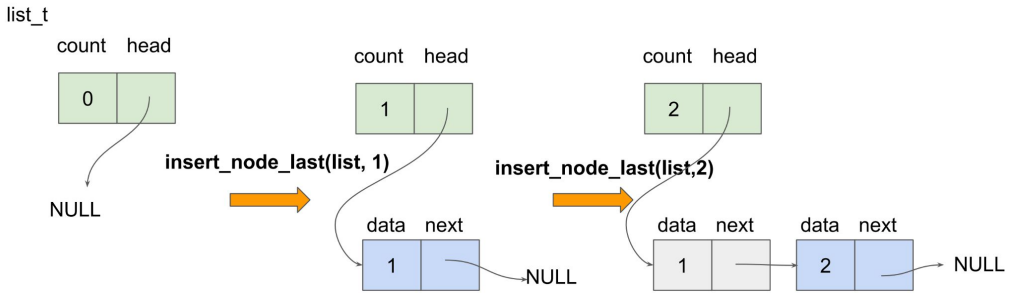
실습 1 - print_list

- Linked list의 모든 노드를 출력하는 함수를 구현해 보세요
  - void print_list(list_t *list)
  - head가 가리키는 node부터 시작해서 각 노드에 연결된 다음 노드(next)를 따라가면서 NULL이 발견될때까지 순회하며 출력
- node_t 내 next를 통해 연결되어있는 다음 노드를 찾을 수 있습니다
  - next 또한 포인터라는점 염두해주세요



실습 1 - insert_node_last

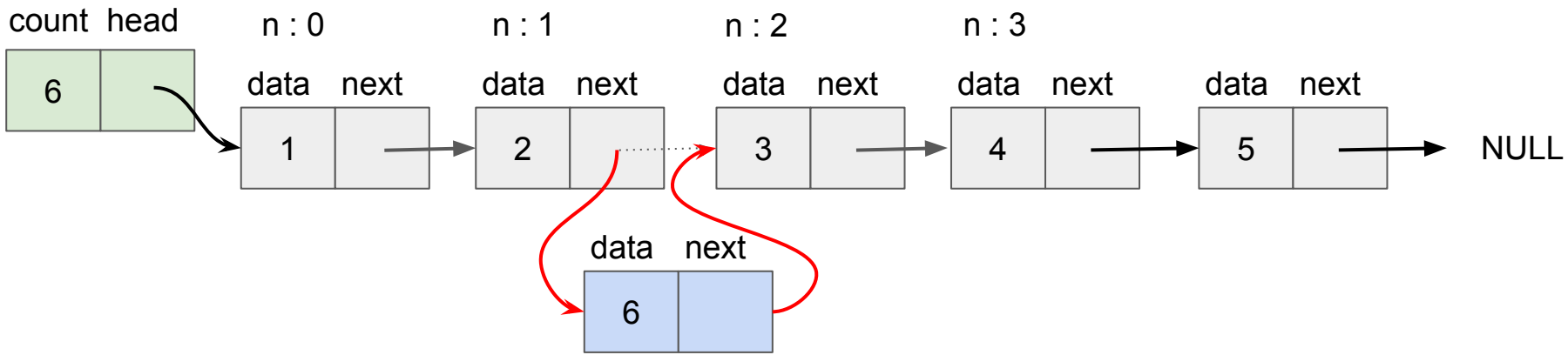
- List의 가장 뒤쪽에 node를 삽입하는 함수를 함께 구현해 봅시다
  - int insert_node_last(list_t *list, int data)
- list의 가장 마지막에 노드를 삽입하는 방법은?
  - 1. 노드를 생성함
  - 2. List상 가장 마지막 위치에 있는 노드를 찾음 (next가 NULL인 노드)
  - 3. 가장 마지막 위치에 있는 노드 뒤에 생성한 노드를 붙여줌



실습 1 - insert_node

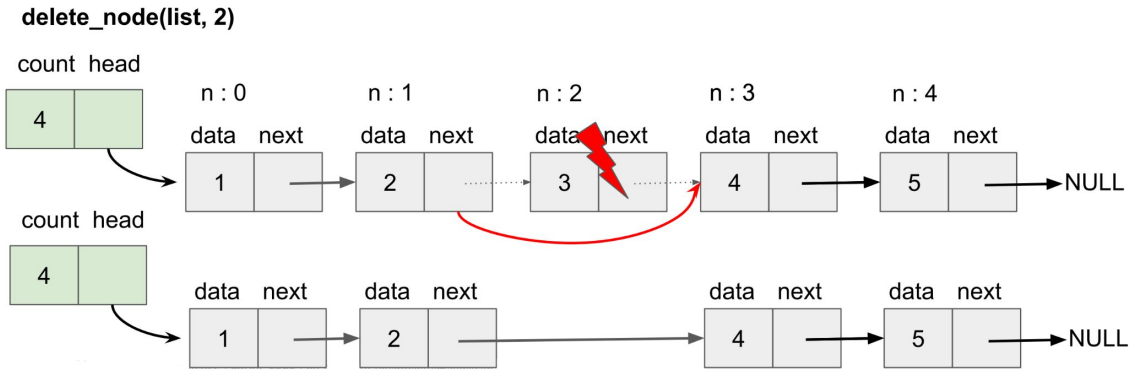
- List 중간에 node를 삽입하는 함수를 구현해보세요
  - int insert_node(list_t *list, int data, int n)
  - n번째 노드 위치에 새로운 노드를 삽입

insert_node(list, 6, 2)



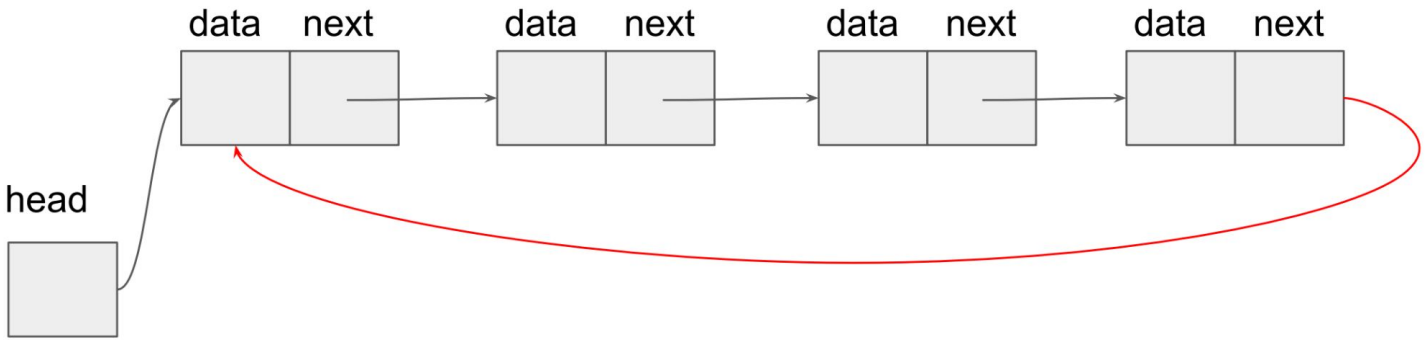
실습 1 - delete_node

- List 중간에 node를 제거하는 함수를 구현해보세요
  - int delete_node(list_t *list, int n)
  - n번째 노드를 지움



실습 2 - Circular linked list

- Circular linked list
  - 마지막 노드가 첫번째 노드를 가리키는 linked list
  - 하나의 노드에서 링크를 계속 따라가다보면 모든 노드를 거쳐 자기 자신으로 돌아옴

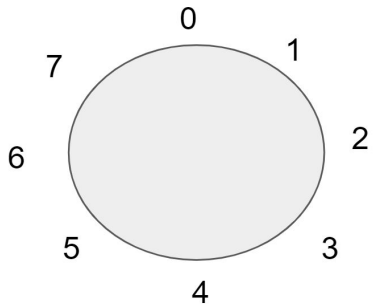


실습 2 - Circular linked list

- Circular linked list를 구현 해 봅시다.
- Singly linked list와 다른점은 제일 첫번째 노드가 변경 될 경우입니다.
  - Singly linked list에서 첫번째 노드를 제거할 경우, head만 옮겨주면 되었지만 circular linked list에서는 가장 마지막 노드(tail)의 next를 첫번째 노드의 next로 설정해주어야합니다.
  - 첫번째 idx에 노드를 삽입 하는 경우에도, 마지막 노드의 next를 수정해야 합니다.

실습 3 - 요세푸스 순열

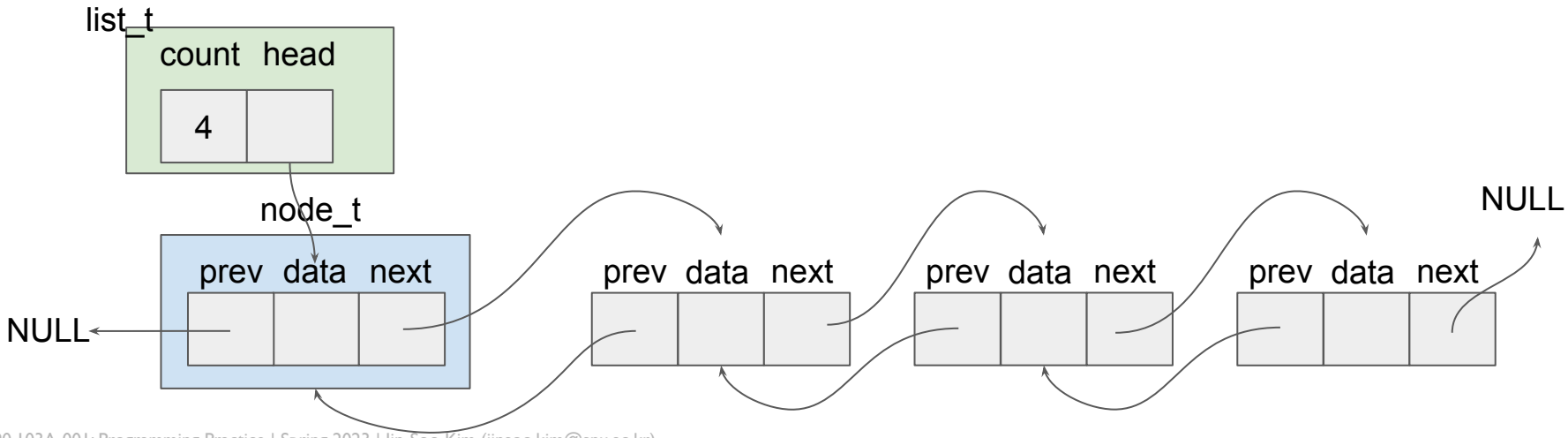
- 요세푸스 순열을 구하는 프로그램을 만들어보세요
  - 0 ~ N-1번까지 N명의 사람이 원을 이루며 앉아있음
  - 순서대로 K번째 사람을 빼고, 한사람이 빠지면 남은 사람들로 이 과정을 계속 해감 (K > 0)
  - N명의 사람이 모두 제거될때까지 계속됨
  - 원에서 사람들이 제거되는 순서를 (N,K) - 요세푸스 순열이라고 함
- 실습 2에서 작성했던 Circular linked list를 사용해보세요



0 1 **2** 3 4 5 6 7
0 1 3 4 **5** 6 7
0 1 3 4 6 7
1 3 **4** 6 7
1 3 6 7
3 6 **7**
3 6
6 => (8,2) : 2 5 0 4 1 7 3 6

과제 2 - Doubly linked list

- Doubly linked list는 이전 노드(prev)를 가리키는 포인터가 추가로 있는 리스트입니다.
- 각 node는 다음 node와 이전 node를 가리키는 포인터가 필요합니다.



실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 다음주 수요일 23:59 까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
- 제출 방법
  - Elice의 Submit 기능 활용