4190.103A-001: Programming Practice Lab. 12

Euidong Lee
Junseok Lee
Minhyo Jeong
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2022

과제 풀이

Lab 11 과제 1

Lab 11 과제1 - 설명

- 원하는 석차의 학생을 찾아내는 프로그램을 만들어 봅세요
- 각 학생 마다 다음의 정보가 주어집니다
  - 학번 - 학생마다 고유, 자연수로만 이루어져 있음
  - 점수 - 자연수로만 이루어져 있음, 동점자 존재
- 학생의 석차는 다음의 규칙으로 결정됩니다
  - 점수가 높은 순서대로 석차를 매깁니다
  - 동점자의 경우, 학번이 작은(앞번호) 가 더 높은 석차를 갖습니다

Lab11 과제1 - 설명

- 입력은 다음과 같이 주어집니다
  - [학생 수] [학생 1의 학번] [학생 1의 점수] [학생 2의 학번] [학생 2의 점수] …
  - [찾아보고 싶은 학생의 석차]
- 출력하셔야 하는 결과는 다음과 같습니다
  - [해당 석차 학생의 학번] [해당 석차 학생의 점수]

```
> make -s
> ./main
5 1 60 2 60 3 78 4 90 5 100
4
1 60
>
```

```
> make -s
> ./main
8 1 60 2 60 3 60 4 60 5 60 6 60 7 60 8 60
3
3 60
>
```

Lab11 과제1 - 풀이

- sort_by_score : 점수가 큰 순으로 함수
- sort_by_id : 점수가 같으면 ID가 작은 순으로 정렬하는 함수

```
int main(void) {
  int num;
  int rank;
  struct student students[MAX];

  scanf("%d", &num);
  for (int i = 0; i < num; i++)
    scanf("%d %d", &students[i].id, &students[i].score);


  sort_by_score(students, num);
  sort_by_id(students, num);

  scanf("%d", &rank);
  printf("%d %d\n", students[rank - 1].id, students[rank
- 1].score);

  return 0;
}
```

Lab11 과제1 - 풀이

- struct copy시 memcpy사용가능
  - memcpy(s1, s2, sizeof(student_t));
- struct copy시 member별로 copy해도 무방
  - s1->id = s2->id;
  - s1->score = s2->score;

```
void swap(student_t * s1, student_t * s2) {
  struct student temp;
  temp = *s1;
  *s1 = *s2;
  *s2 = temp;
}

void sort_by_score(student_t * students, int num) {
  for (int i = 0; i < num - 1; i++)
    for (int j = i + 1; j < num; j++)
      if (students[i].score < students[j].score)
        swap(&students[i], &students[j]);
}

void sort_by_id(student_t * students, int num) {
  for (int i = 0; i < num - 1; i++)
    for (int j = i + 1; j < num; j++)
      if ((students[i].score == students[j].score) &&
          (students[i].id > students[j].id))
        swap(&students[i], &students[j]);
}
```

Lab11 과제1 - 학생답안

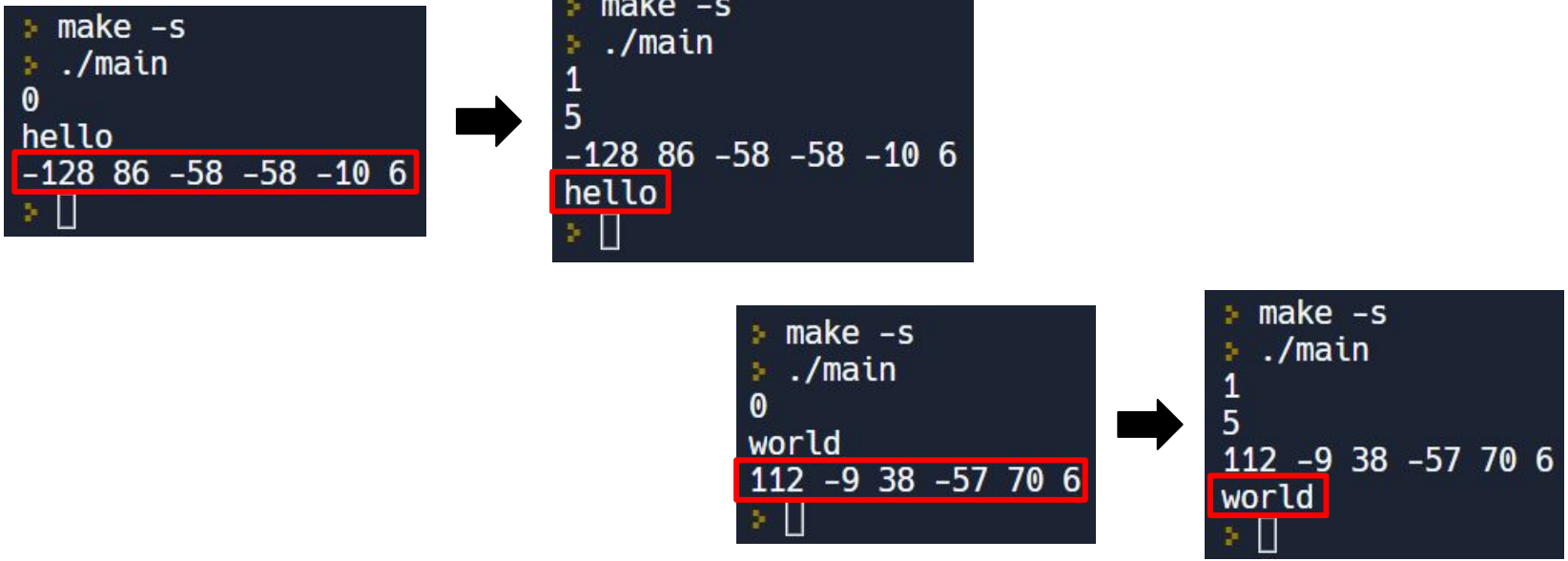
```c
void sort(int *grade, int *grade_next, int
*num, int *num_next, int *rank){
  int count = 0;
  //int gn = *grade_next;
  //int g = *grade;
  //printf("%d %d\n", gn, g);
  if(*grade_next > *grade){
    count++;
  }
  else if(*grade_next == *grade){
    if(*num > *num_next){
      count++;
    }
  }
  *rank = *rank + count;
}
```

```c
int main(void) {
  struct students st[100];
  int n;

  scanf("%d", &n);

  for(int i = 0; i < n; i++){
    scanf("%d", &st[i].num);
    scanf("%d", &st[i].grade);
    st[i].rank = 1;
  }
  for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      //printf("I: %d j: %d  ", i, j);
      sort(&st[i].grade, &st[j].grade, &st[i].num,
&st[j].num, &st[i].rank);
      //for(int k = 0; k < n; k++)
        //printf("%d %d /", st[k].rank, st[k].grade);
      //printf("\n");
    }
  }
```

Lab 11 과제 2

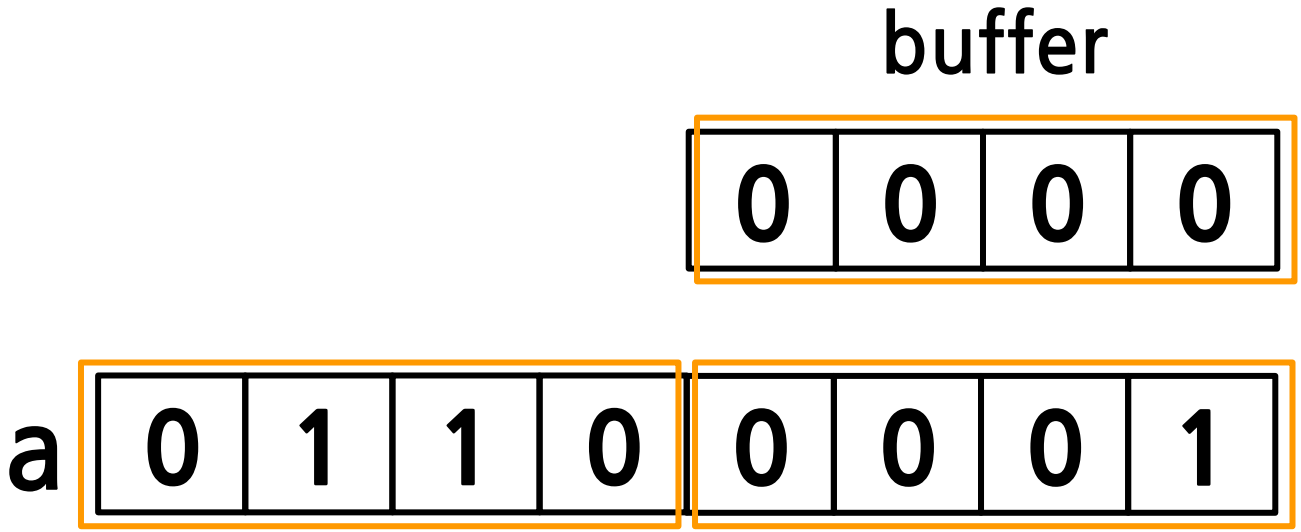
Lab11 과제2 - 설명

- Bitwise operation을 이용해서 암호/복호 프로그램을 만들어봅시다!

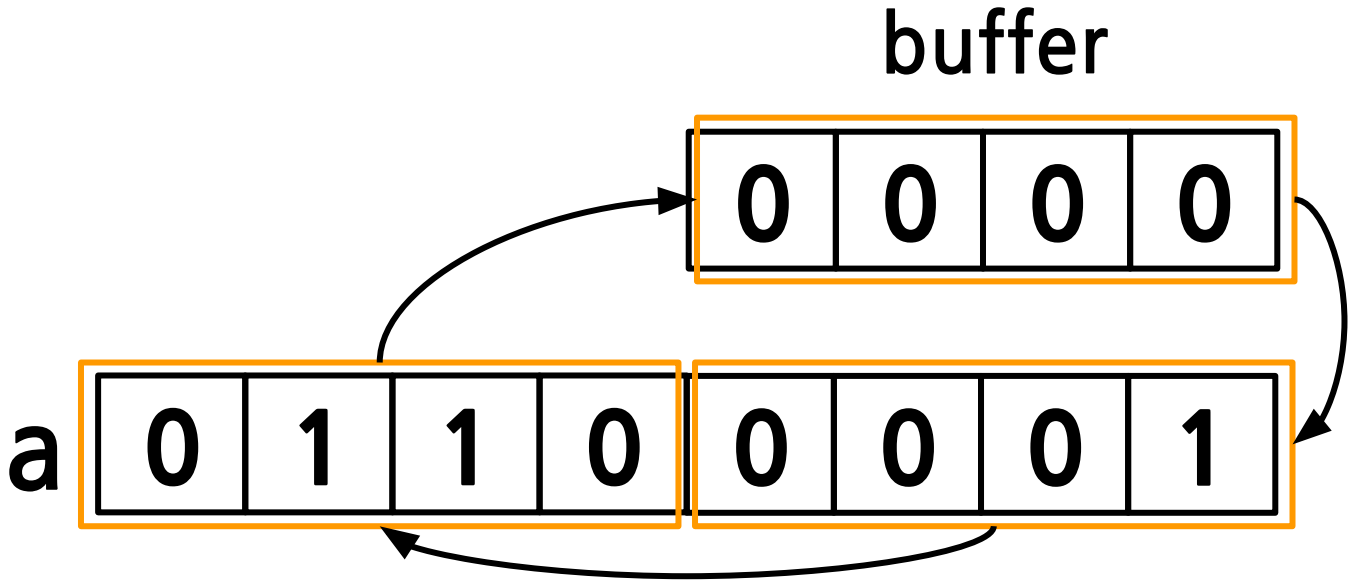


Lab11 과제2 - 설명

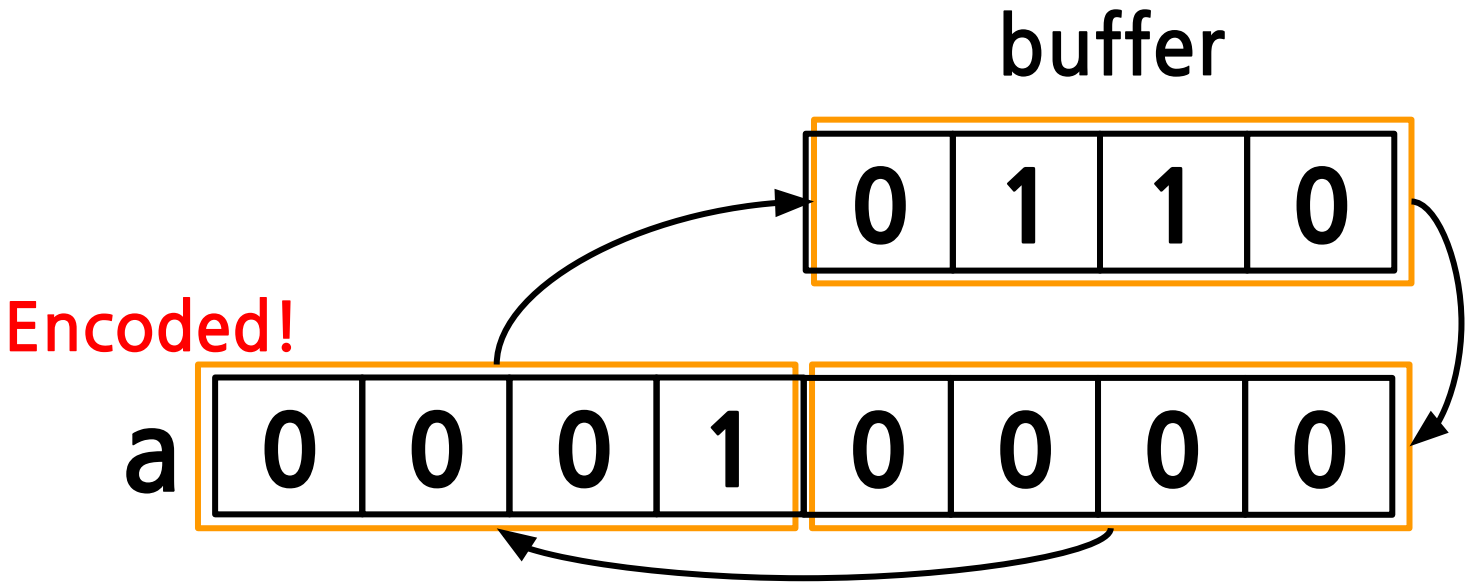
Lab11 과제2 - 설명



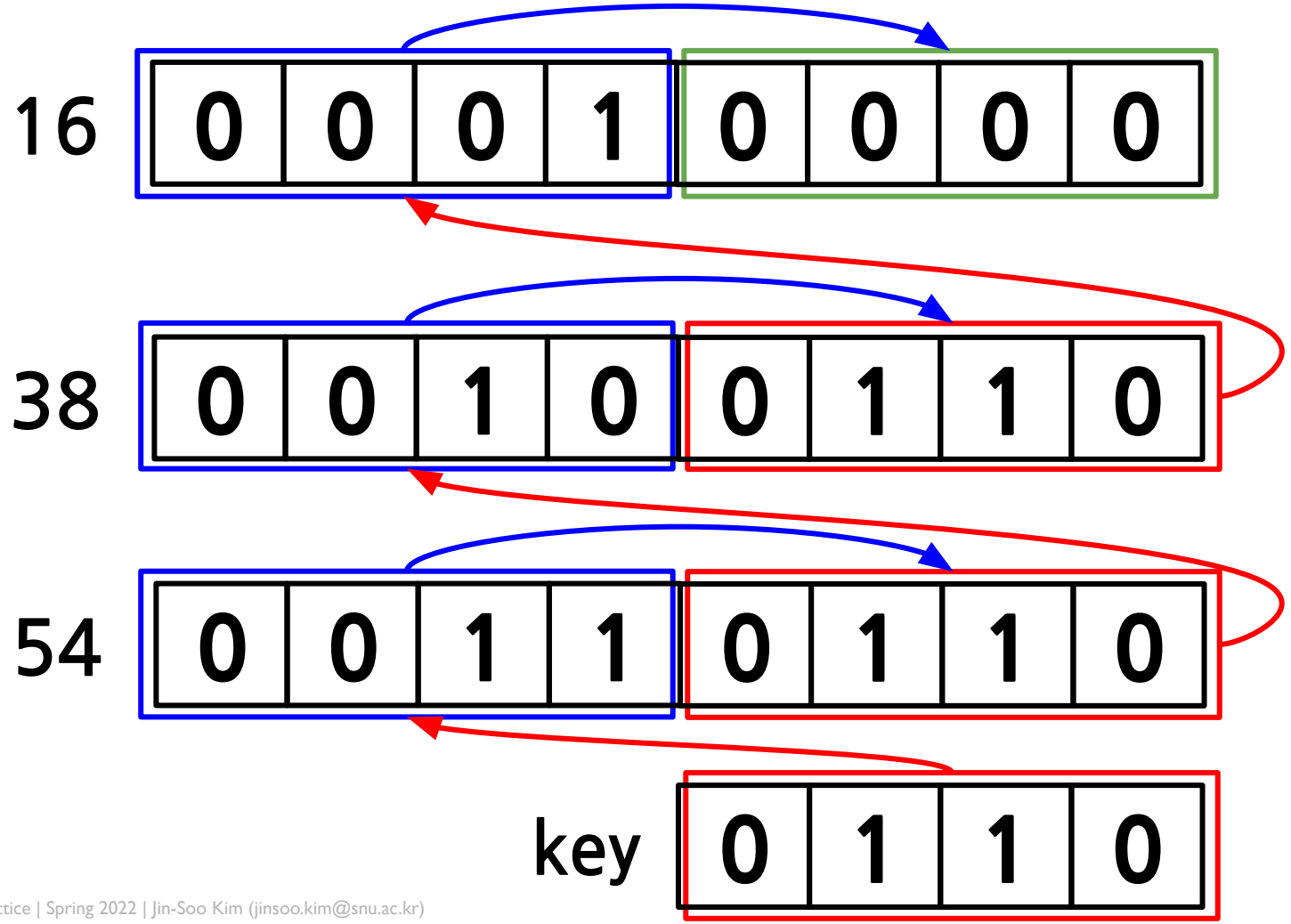
Lab11 과제2 - 설명



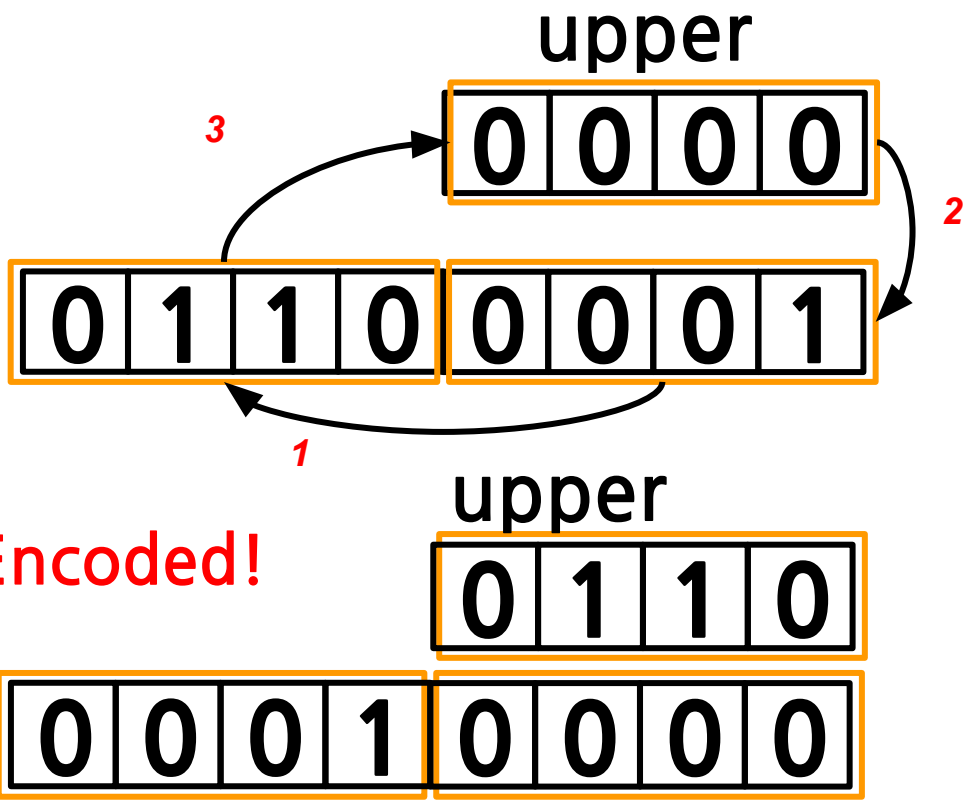
Lab11 과제2 - 설명



Lab11 과제2 - 설명

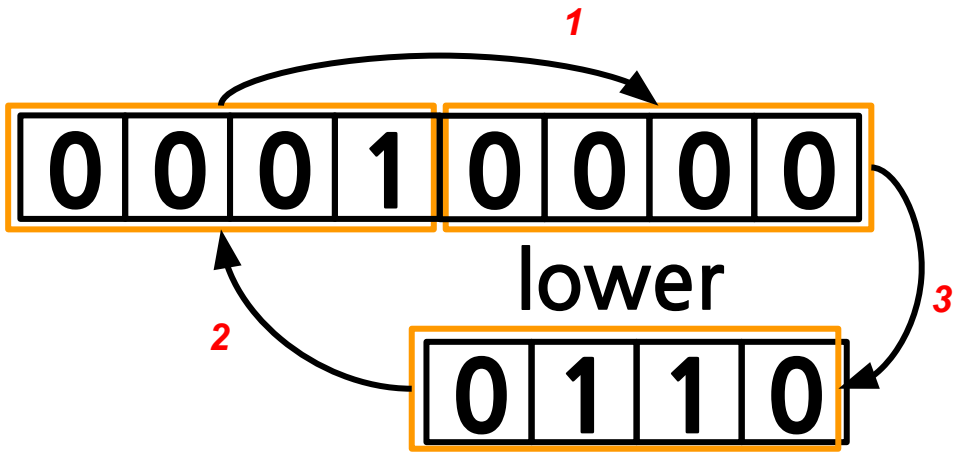


Lab11 과제2 - 풀이

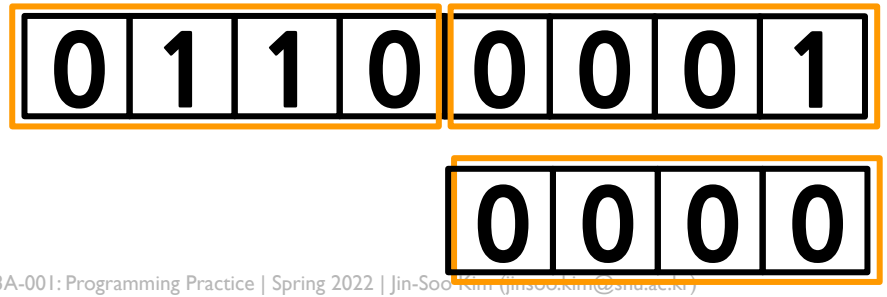



```
void encode(char *str, int length){
  char encoded, upper = 0;
  for(int i = 0; i < length; i++){
1   encoded = str[i] << NUM_SHIFT;
2   encoded |= upper;
3   upper = str[i] >> NUM_SHIFT;
    printf("%d ", encoded);
  }
  printf("%d\n", upper);
}
```

Lab11 과제2 - 풀이



Decoded!




```
void decode(int *code, int length, int
key){
  char result[MAX] = {0,};
  char decoded, lower = key & BIT_MASK;

  for(int i = length-1; i >= 0; i--){
  1 decoded = (code[i] >> NUM_SHIFT)
                                & BIT_MASK;
  2 decoded |= (lower << NUM_SHIFT);
  3 lower = code[i] & BIT_MASK;
    result[i] = decoded;
  }
  result[length] = '\0';
  printf("%s\n", result);
}
```

Lab11 과제2 - 풀이

```c
int main(){
  int option, c;
  int len, key, code[100];
  char str[MAX];
  scanf("%d", &option);
  switch(option){
    case 0:
      scanf("%s", str);
      len = strlen(str);
      encode(str, len);
      break;

    case 1:
      scanf("%d", &len);
      for (int i = 0; i < len; i++)
        scanf("%d", &code[i]);
      scanf("%d", &key);
      decode(code, len, key);
  }
  return 0;
}
```

Lab11 과제2 - 학생답안

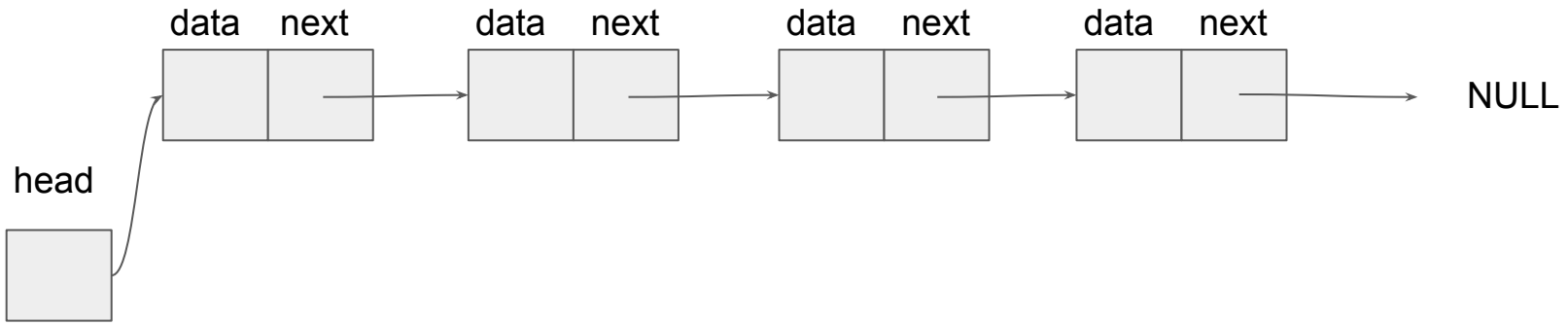
```c
int main(void) {
  int option;
  scanf("%d", &option);
  /*encoding*/
  if(option == 0) {
    char text[MAX];
    scanf("%s", text);

    for(int i = 0; i < strlen(text); i++){
      if(i==0)
        printf("%d ", (char)(text[i] << 4));
      else
        printf("%d ", (char)((text[i] << 4) +
(text[i-1] >> 4)));
    }
    printf("%d\n", text[strlen(text)-1] >> 4);
  }
```

```c
/*decoding*/
  else if(option == 1) {
    int num; //total of numbers to be decoded
    scanf("%d", &num);

    int decode[num+1];
    for(int i = 0; i < num+1; i++)
      scanf("%d", &decode[i]);

    unsigned char decode_txt[num+1];
    for(int i = 0; i < num+1; i++)
      decode_txt[i] = decode[i];

    for(int i = 0; i < num; i++)
      printf("%c", (decode_txt[i] >> 4) +
(decode_txt[i+1] << 4));
    printf("\n");
  }
```

실습 1

실습 1 – 설명

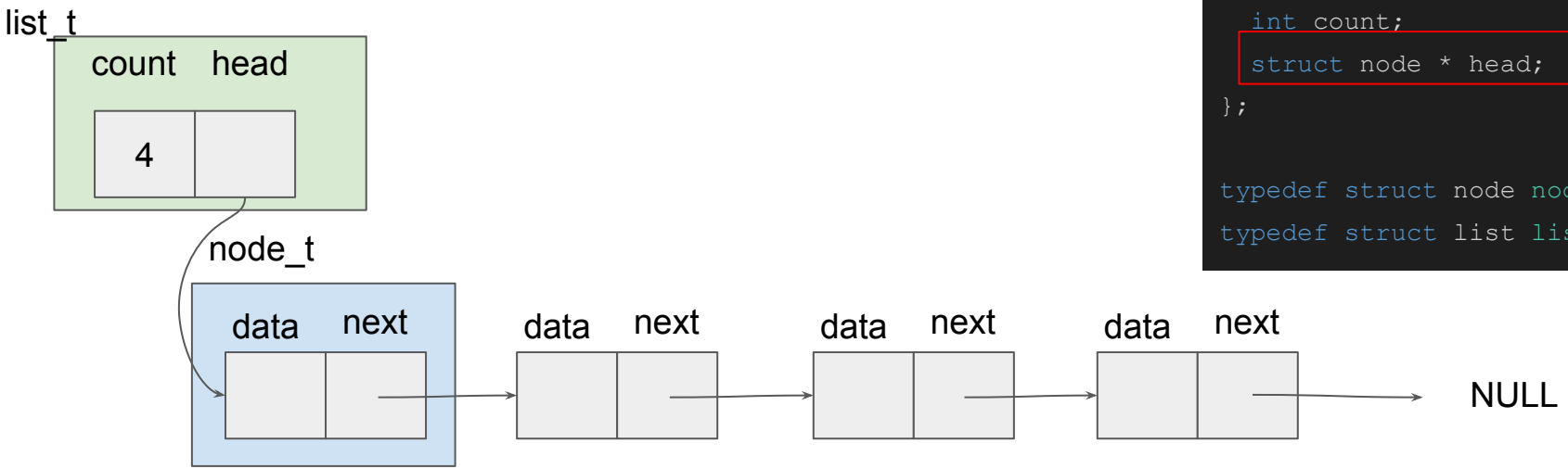
- Singly linked list
  - Element간의 연결을 통해 리스트를 구현한 자료구조
  - 노드들이 한쪽 방향으로 만 연결이 되어있음
- Interface
  - insert_node_first(), insert_node_last(), insert_node(), delete_node(), print_list(), etc…



실습 1 – 설명

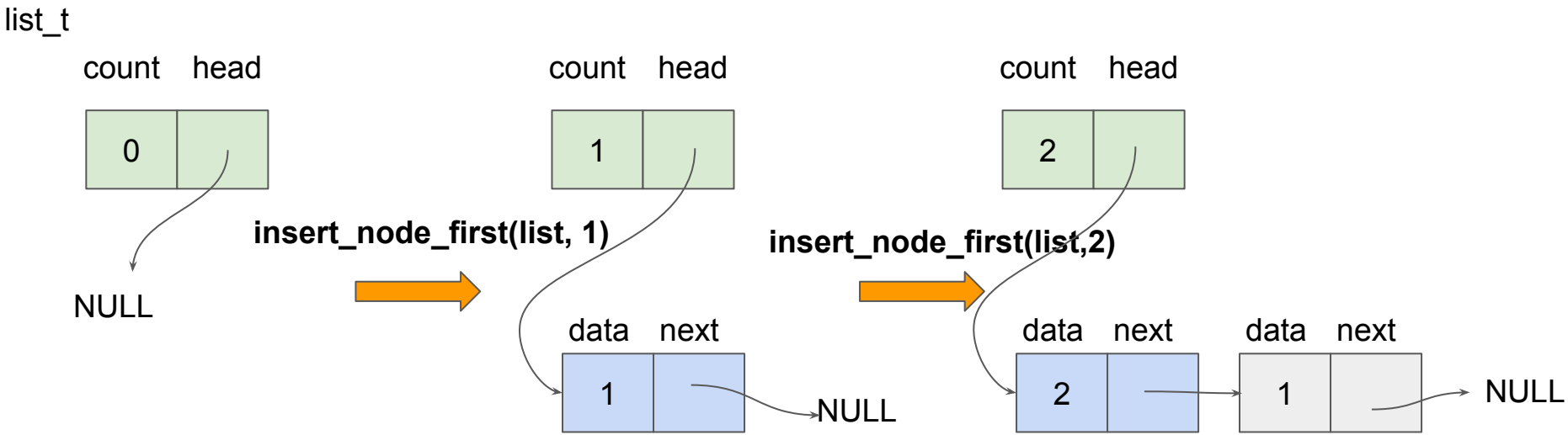
- Singly linked list를 위한 structure를 정의해봅시다.
- 각 node는 다음 node를 가리키는 포인터가 필요합니다.
- List의 첫번째 노드를 가리키는 포인터 또한 필요합니다.

```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;
```



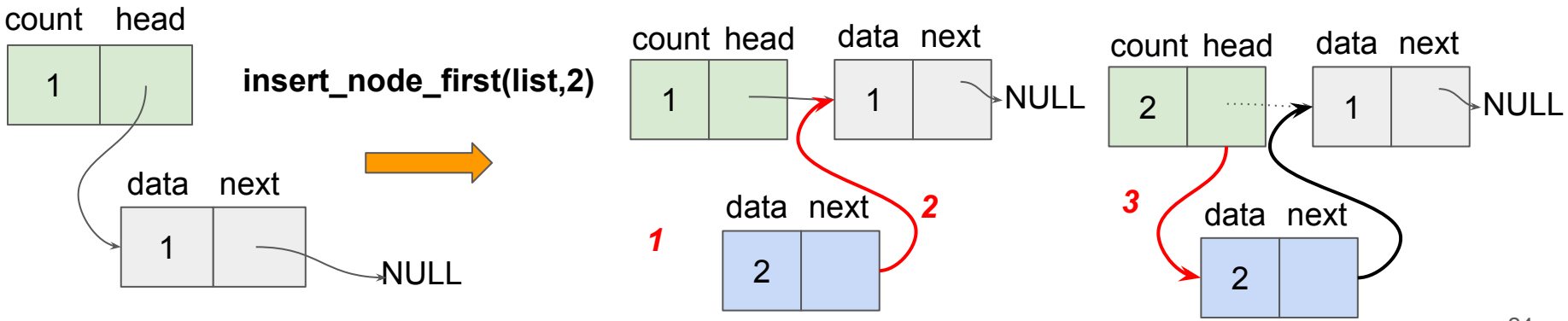
실습 1 – insert_node_first()

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

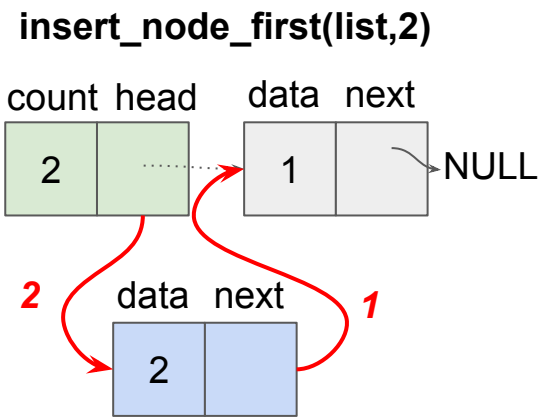


실습 1 – insert_node_first()

- 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



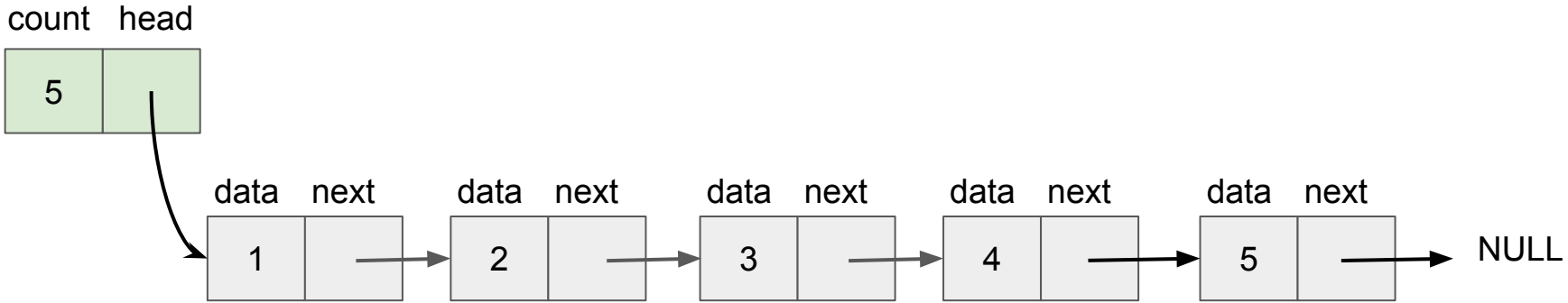
실습 1
– insert_node_first()




```c
int insert_node_first(list_t * list, int data) {
  node_t * n = malloc(sizeof(struct node));// 노드 생성


  if (n == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  n->data = data; //생성한 노드의 data부분을 적음
1 n->next = list->head; //생성한 노드를 head가 가리키는 노드에 연결
2 list->head = n; // head가 가리키는 노드를 생성한 노드로 변경
  list->count++; //list의 element count증가


  printf("Insert node front %d\n", data);
  return 1;
}
```

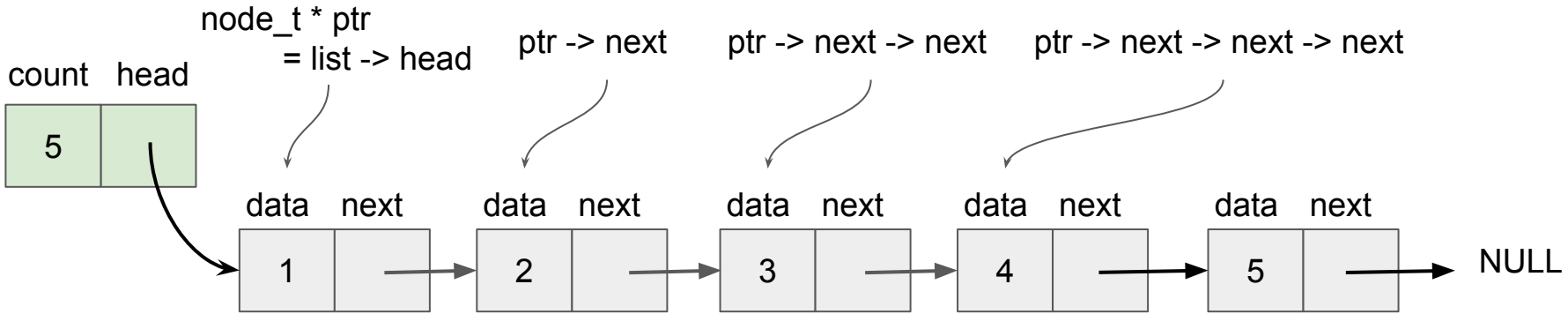
실습 1 – print_list()

- 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



실습 1 – print_list()

- node_t 내 next를 통해 연결되어있는 다음 노드를 찾을 수 있습니다
  - node_t pointer –> next
  - next 또한 포인터라는점 염두해주세요



실습 1 – print_list()

- print_list 함수를 완성해주세요

```
void print_list(list_t * list)
{
  node_t * ptr = /*write your code*/;

  printf("# elements: %d\n", /*write your code*/);
  printf("Data: ");

  /* write your code
     linked list의 모든 노드를 순회하며 data를 출력함
     hint : ptr이 NULL을 가리키면 linked list의 마지막 노드임
  */

  printf("\n");
}
```

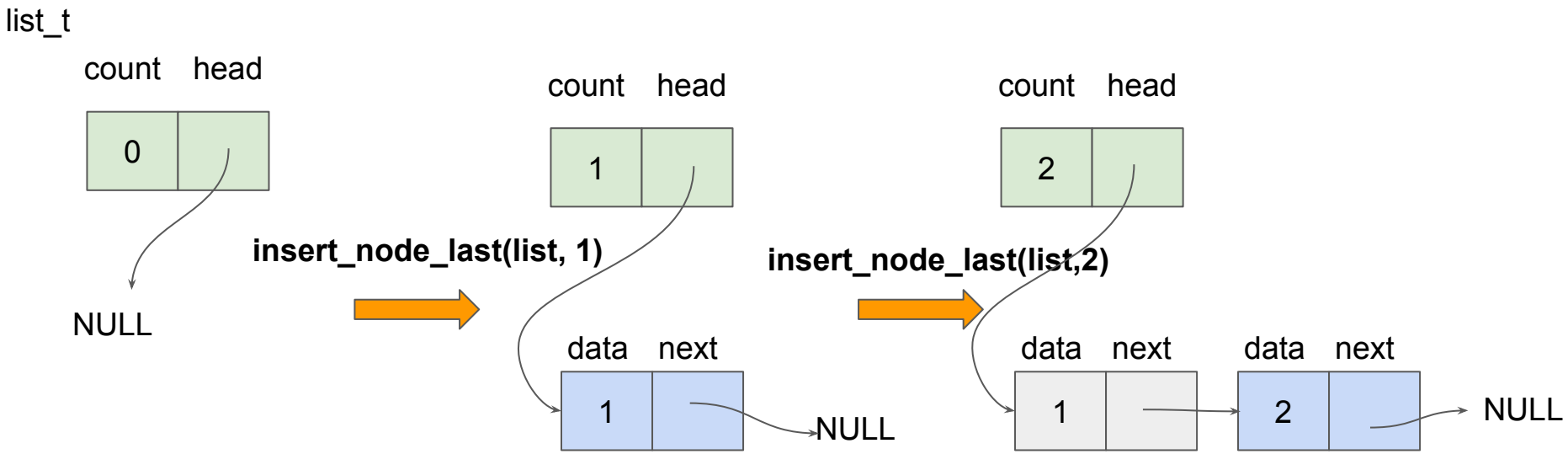
```
Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  c: clear list
input command: f
input data: 10
Insert node front 10
# elements: 1
Data: 10

Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  c: clear list
input command: f
input data: 20
Insert node front 20
# elements: 2
Data: 20 10
```

실습 1 – 풀이

```c
void print_list(list_t * list)
{
  node_t * ptr = list->head;

  printf("# elements: %d\n", list->count);
  printf("Data: ");

  while(ptr != NULL)
  {
    printf("%d ", ptr->data);
    ptr = ptr->next;
  }

  printf("\n");
}
```

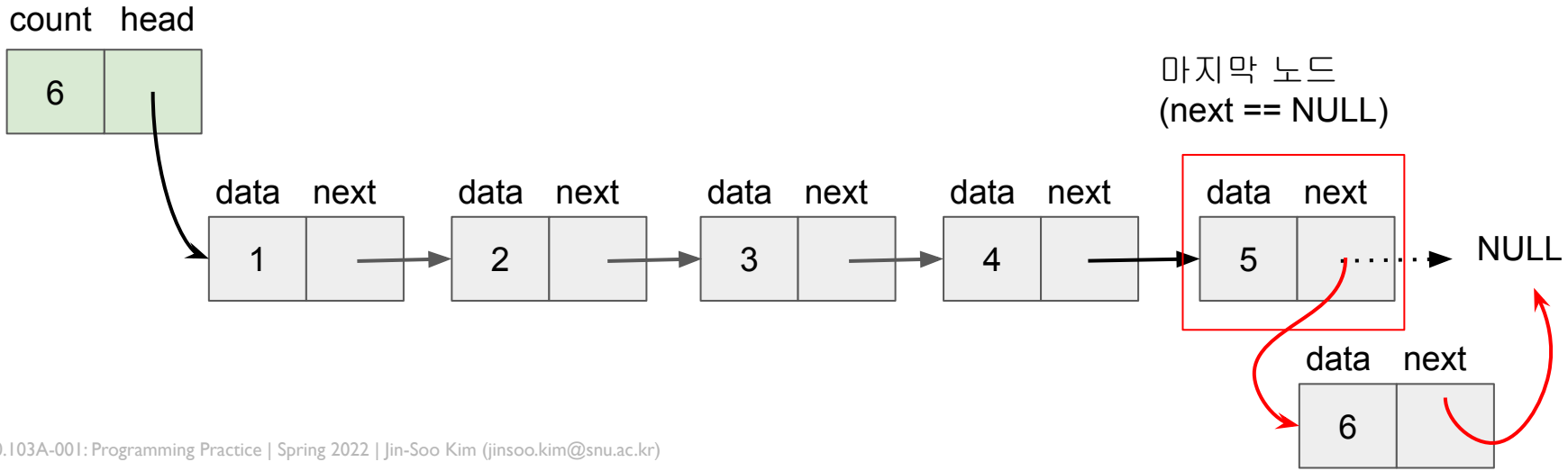
실습 1 – insert_node_last()

- List의 가장 뒤쪽에 node를 삽입하는 함수를 함께 구현해 봅시다
  - int insert_node_last(list_t list)



실습 1 – insert_node_last()

- 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
  - list에 아무 노드도 없었다면, head에 생성한 노드를 연결해주기만 하면 됨



실습 1 – insert_node_last()

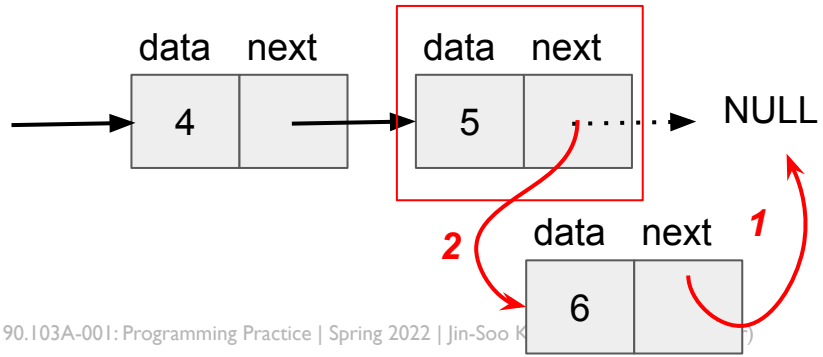
- insert_node_last 함수를 완성해주세요

```
Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  c: clear list
input command: l
input data: 10
Insert node last 10
# elements: 1
Data: 10

Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  c: clear list
input command: l
input data: 20
Insert node last 20
# elements: 2
Data: 10 20
```

```
int insert_node_last(list_t * list, int data) {
  node_t * new_node = malloc(sizeof(struct node));
  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  if (list->head == NULL) {
      /*write your code*/
  }
  else {
    /* write your code
       마지막 노드를 찾아서 생성한 노드와 연결함 */
  }

  printf("Insert node last %d\n", data);
  return 1;
}
```

실습 1 – insert_node_last() – 풀이

```
int insert_node_last(list_t * list, int data)
{
  node_t * new_node = malloc(sizeof(struct node));

  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }
  new_node->data = data;
```

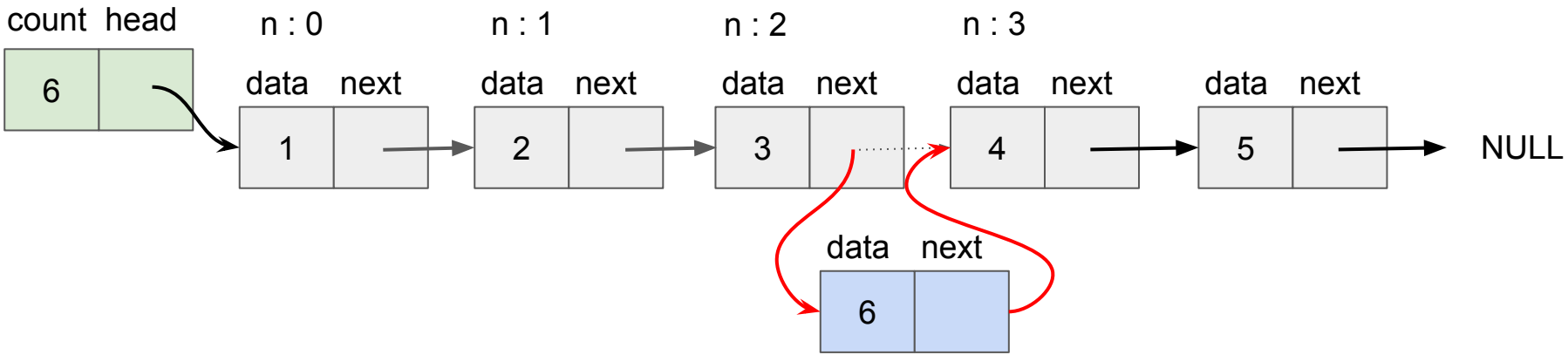



```
  if (list->head == NULL) {
    new_node->next = NULL;
    list->head = new_node;
  }
  else {
    node_t * ptr = list->head;
    while((ptr->next) != NULL) // 가장 마지막 노드를 찾음
      ptr = ptr->next;

1 new_node->next = NULL;
2 ptr->next = new_node;
  }

  list->count++;

  printf("Insert node last %d\n", data);
  return 1;
}
```

실습 2 – insert_node()

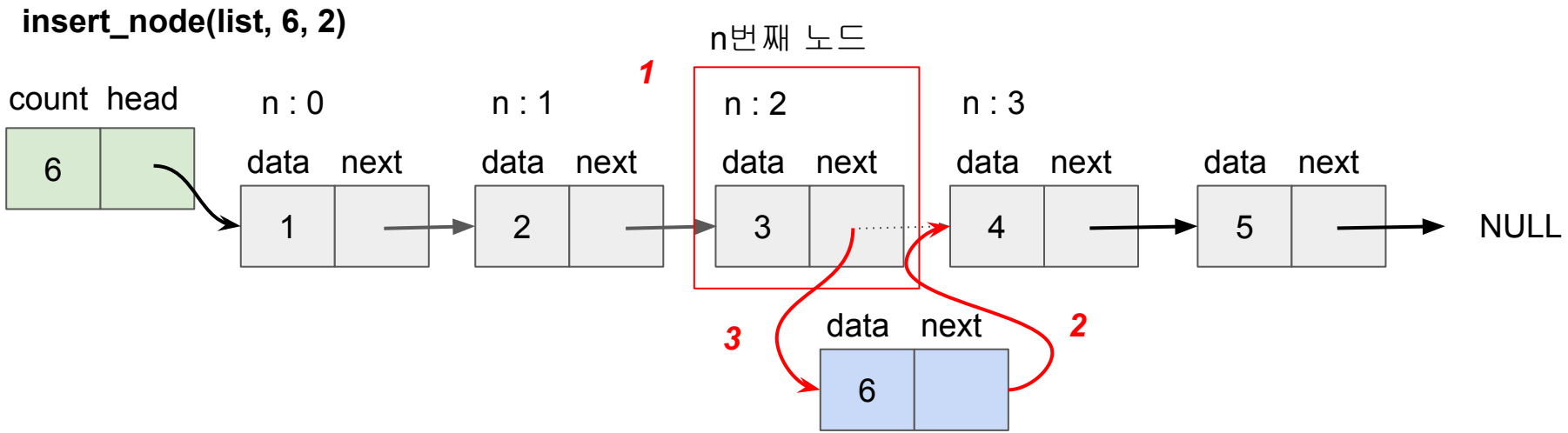
- List 중간에 node를 삽입하는 함수를 구현해보세요
  - int insert_node(list_t *list, int data, int n)
  - n번째 노드 다음에 새로운 노드를 삽입

insert_node(list, 6, 2)



실습 2 – insert_node()

- n번째 노드 뒤에 삽입하는 방법은?
  - 1. n번째 노드를 찾음
  - 2. 생성한 노드의 next를 n번째 노드의 next로 연결함
  - 3. n번째 노드의 next를 생성한 노드로 연결함



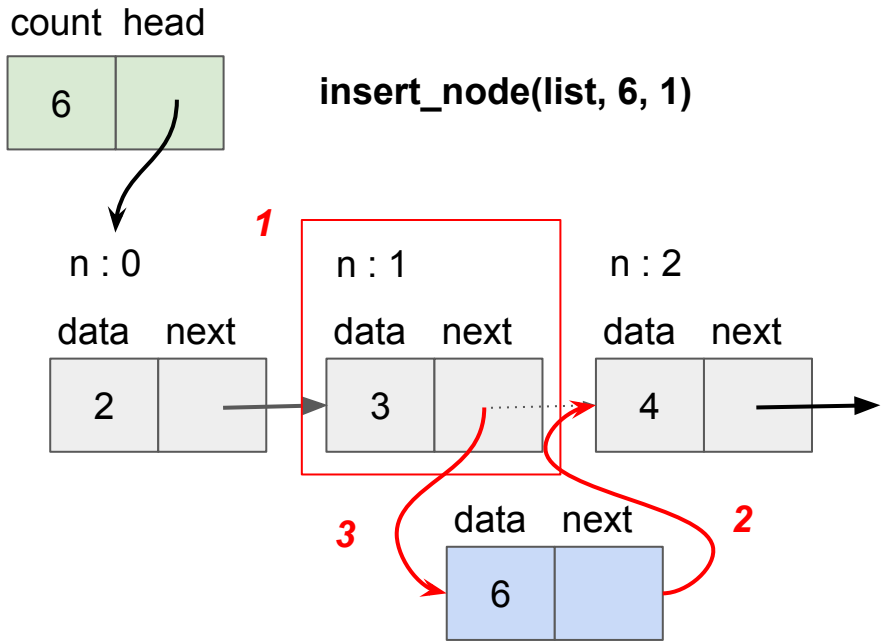
실습 1 – insert_node()

- inser_node 함수를 완성해주세요

```
# elements: 2
Data: 10 20

Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  c: clear list
input command: i
input data, pos: 15 0
# elements: 3
Data: 10 15 20

Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  c: clear list
input command: i
input data, pos: 17 1
# elements: 4
Data: 10 15 17 20
```

```
int insert_node(list_t * list, int data, int n) {
  if (/*write your code*/) // n번째노드가 없을경우
    return 0;

  node_t * new_node = malloc(sizeof(struct node));
  node_t * cur = list->head;

  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  /* write your code
     n번째 노드를 찾음. cur가 해당 노드를 가리키게함 */

  /* write your code
    cur가 가리키는 노드 뒤에 생성한 노드 삽입
    new_node에 data도 씀 */

  return 1;
}
```

실습 1 – insert_node()

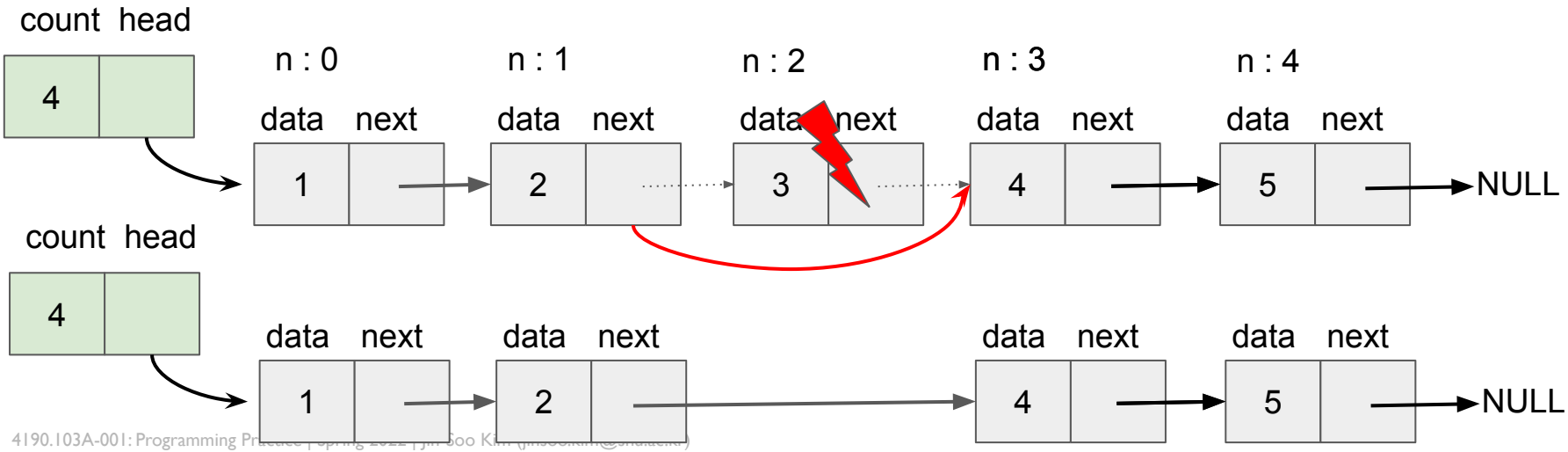



```c
int insert_node(list_t * list, int data, int n){
  if (n >= list->count)// n번째노드가 없다면
    return 0;

  node_t * new_node = malloc(sizeof(struct node));
  node_t * cur = list->head;

  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

1
  for (int i = 0; i < n; i++)
    cur = cur->next;

  new_node->data = data;
2 new_node->next = cur->next;
3 cur->next = new_node;

  list->count++;
  return 1;
}
```

실습 1 – delete_node()

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

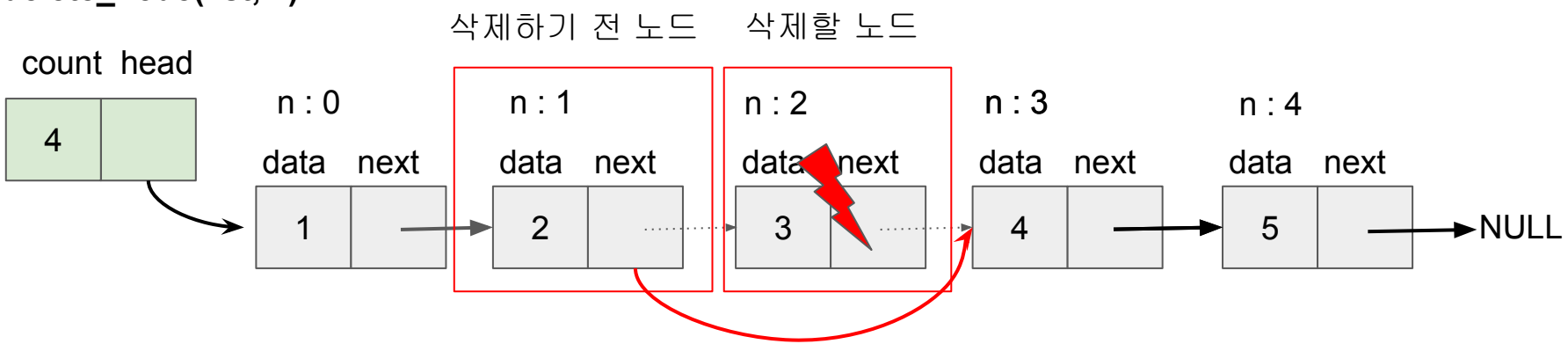
delete_node(list, 2)



실습 1 – delete_node()

- n번째 노드를 삭제하는 방법은?
  - n-1번째 노드와 n번째 노드를 찾음
  - n-1번째 노드의 next를 n번째 노드의 next로 변경하여 리스트상 n번째 노드의 링크를 끊음
  - 제거한 후 list에 아무 노드도 존재하지 않다면, head를 NULL로 변경

delete_node(list, 2)



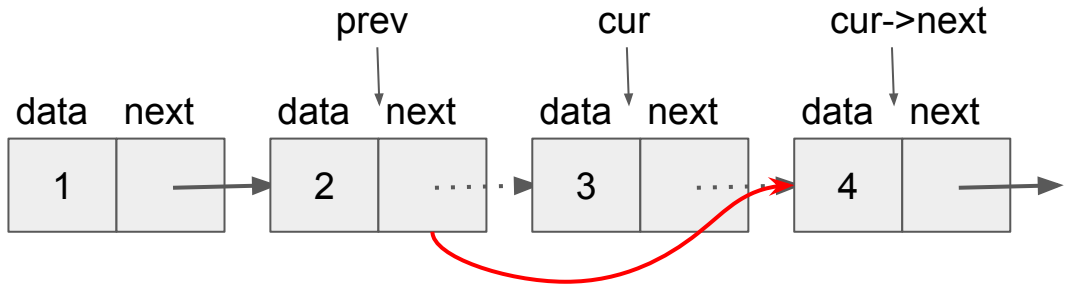
실습 1 – delete_node()

- delete_node 함수를 완성해주세요

```
Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  r: reverse list
input command: f 20
input data: Insert node front 20
# elements: 3
Data: 20 30 40

Command list
f: insert first, l: insert last, i: insert
d: delete,       p: print list,  r: reverse list
input command: d
input pos: 1
Delete node 30
# elements: 2
Data: 20 40
```

```
int delete_node(list_t * list, int n) {
  if (/*write your code*/) // n번째노드가 없을경우
    return -1;
  node_t * cur = list->head;
  node_t * prev = NULL;
  int data;
  /* write your code
    cur가 pos번째 노드가 될때까지 링크를 따라감
    prev도 함께 update하여 prev노드를 관리함
  */

  /* write your code
     제일 첫번째 노드를 지우면 head를 변경함
     제일 첫번째 노드가 아니면 prev의 next를 cur의 next에
     연결하여 cur 노드를 리스트에서 끊어줌
  */

  data = cur->data;
  free(cur);
  printf("Delete node %d\n", data);
  return data;
}
```

실습 1 – delete_node()

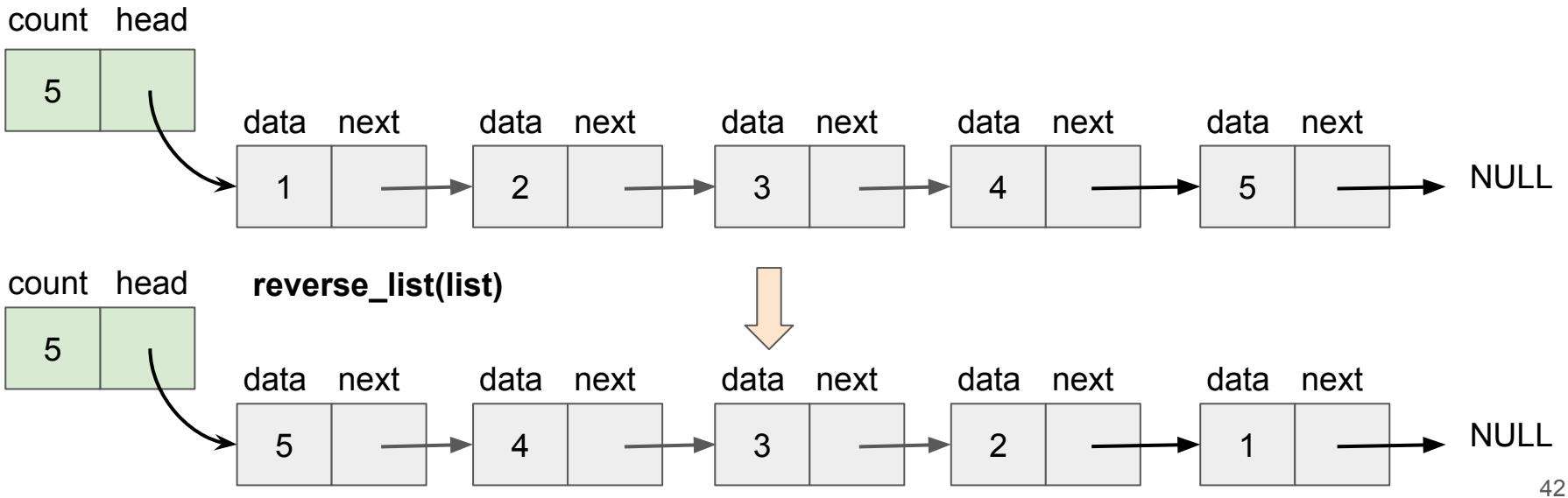



```c
int delete_node(list_t * list, int n) {
  if (n >= list->count)
    return -1;

  node_t * cur = list->head;
  node_t * prev = NULL;
  int data;

  for (int i = 0; i < n; i++) {
    prev = cur;
    cur = cur->next;
  }
```

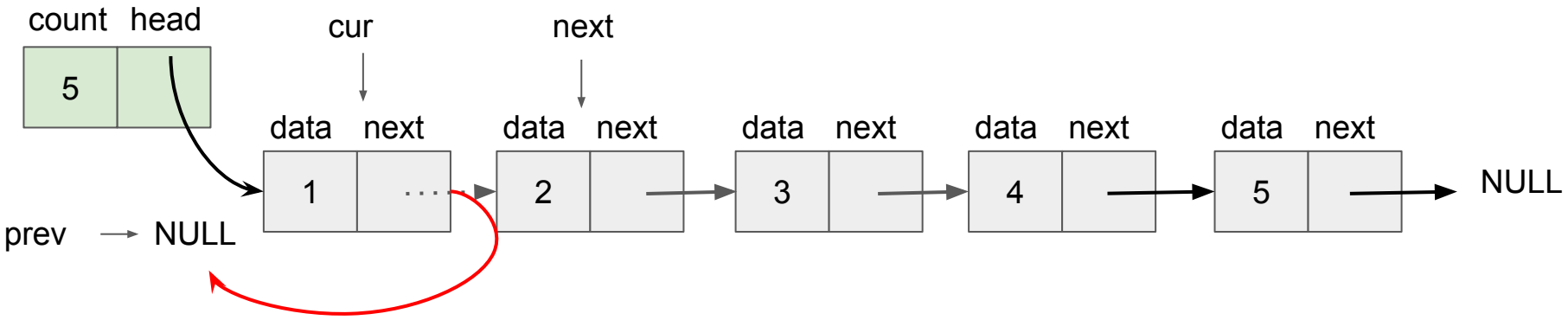
```c
  if (cur == list->head)
    list->head = cur->next;
  else
    prev->next = cur->next;

  list->count--;
  data = cur->data;
  free(cur);

  printf("Delete node %d\n", data);
  return data;
}
```

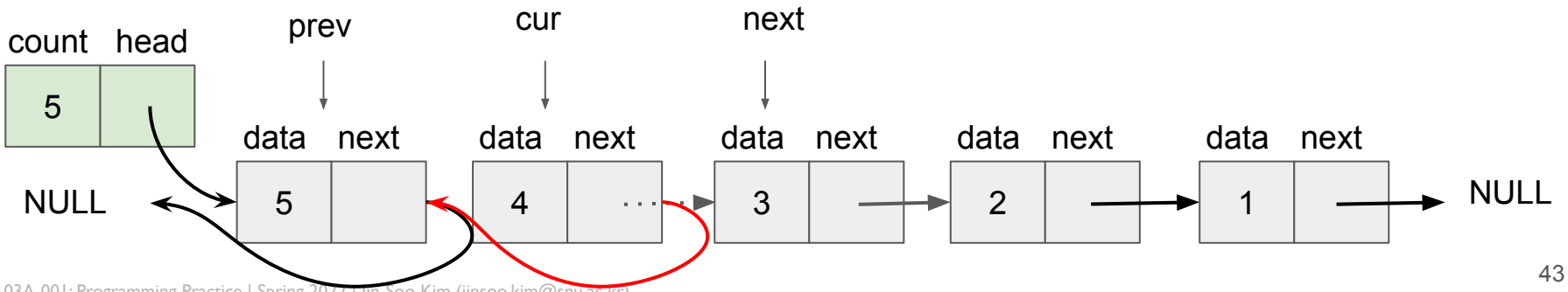
실습 1 – reverse_list() – **제출**

- Linked list를 반대로 뒤집어서 순서를 바꾸는 함수를 구현해보세요
  - void reverse_list(list_t *list)
  - 여러가지 방법으로 생각해보세요

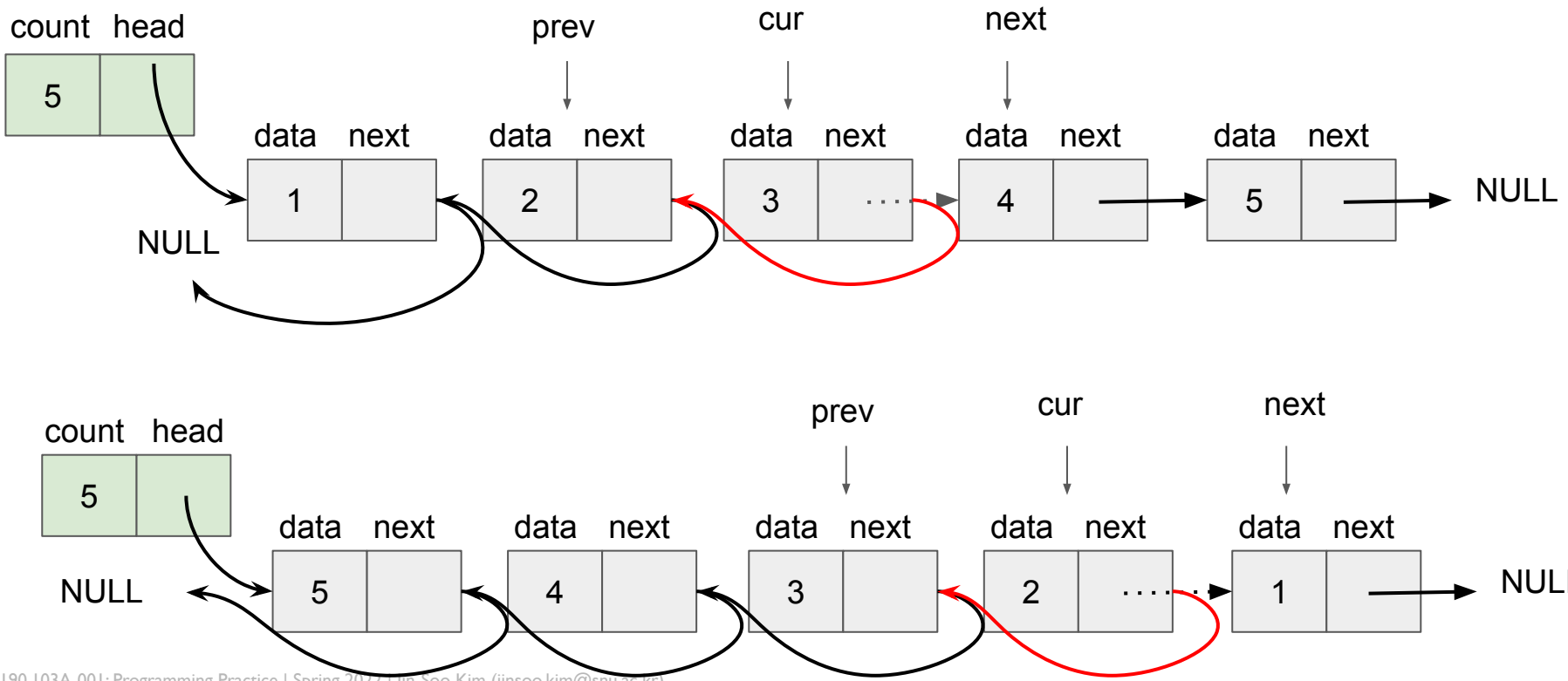


실습 1 – reverse_list() – **제출**

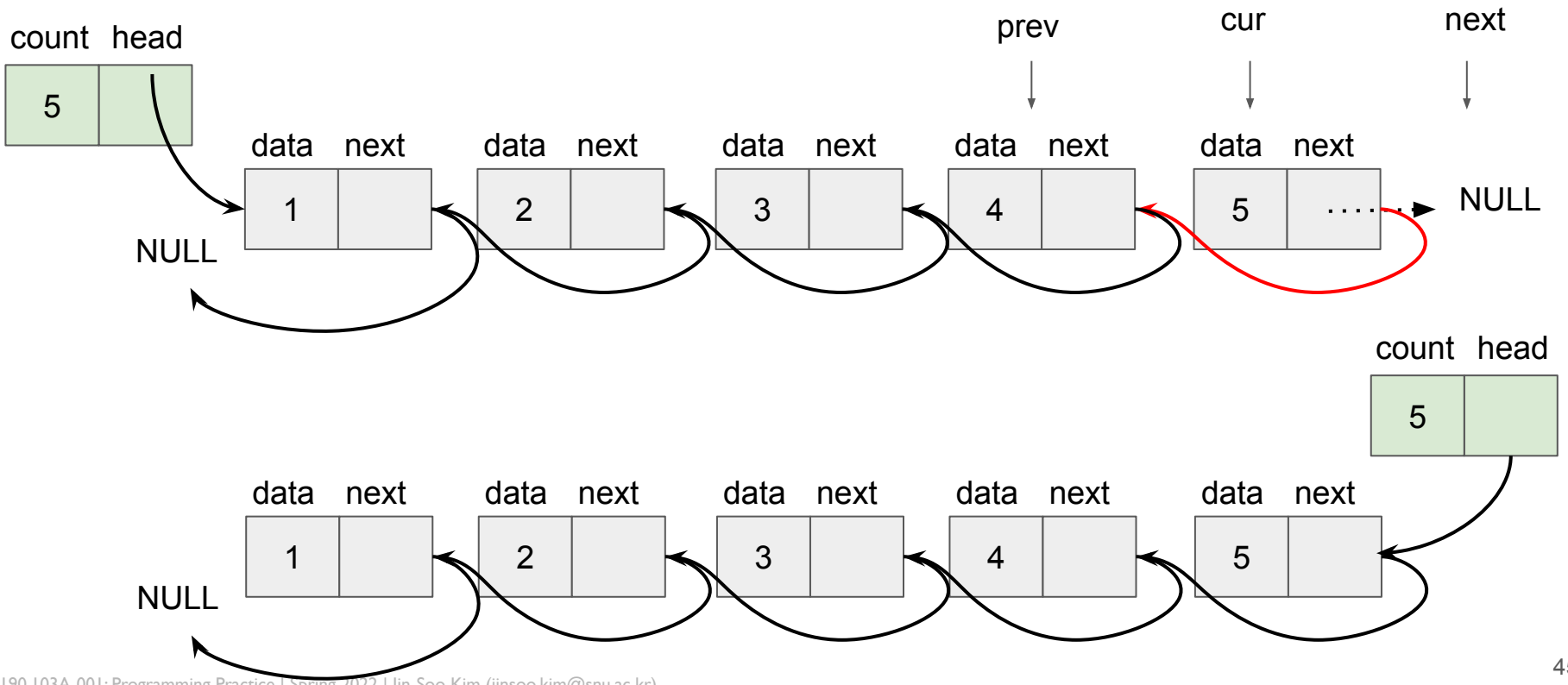




실습 1 – reverse_list() – **제출**

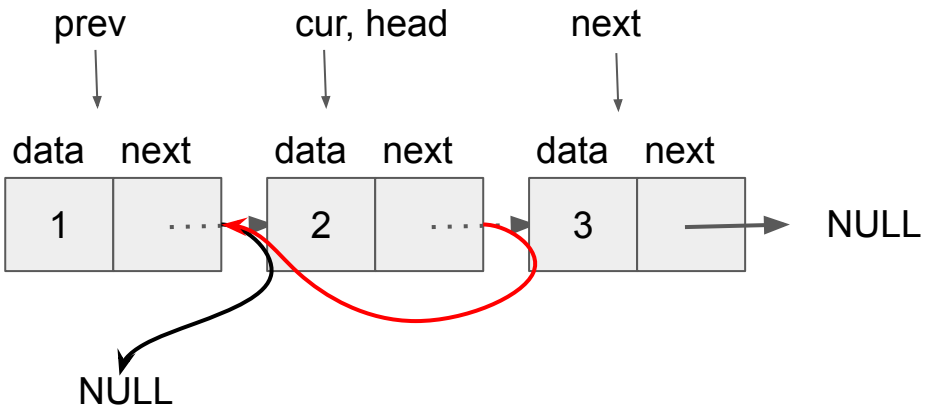


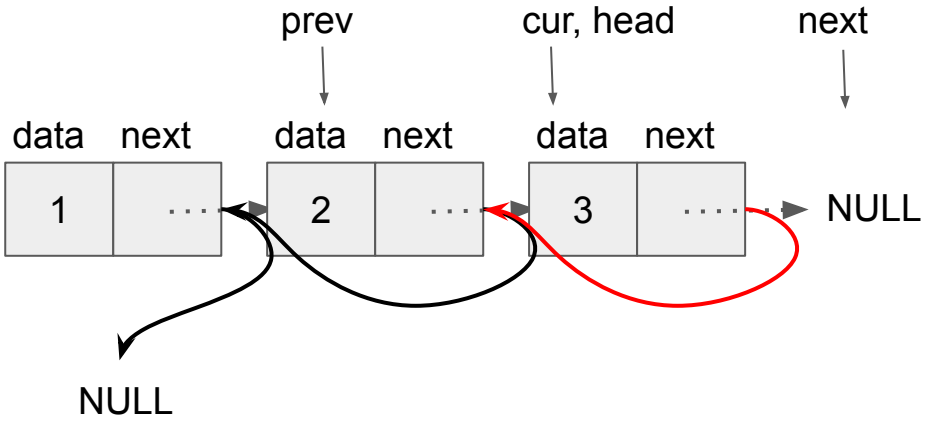
실습 1 – reverse_list() – **제출**



실습 1 – reverse_node()

```c
int reverse_list(list_t * list)
{
  node_t * cur = list->head;
  node_t * prev = NULL;

  while (cur != NULL) {
    node_t * next = cur->next;
    cur->next = prev;
    prev = cur;
    list->head = cur;
    cur = next;
  }
  return 1;
}
```

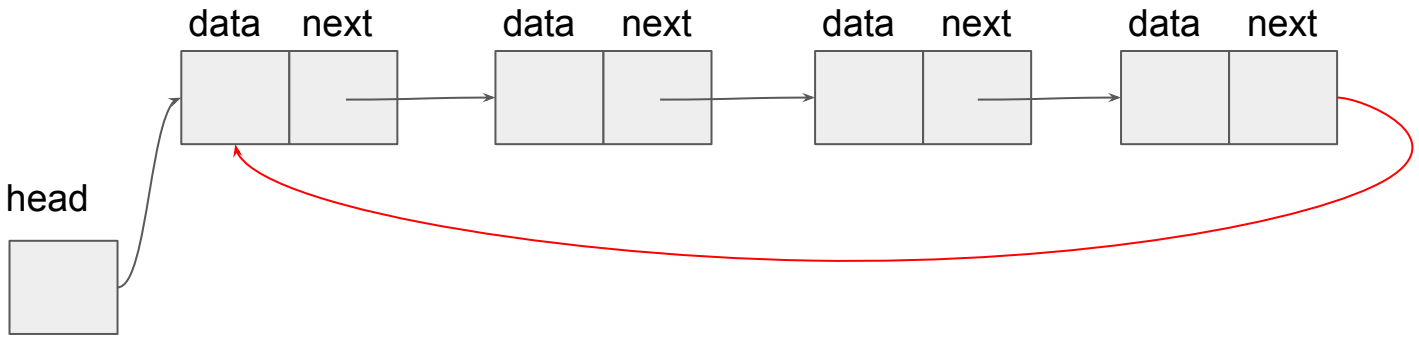




실습 2

실습 2 – 설명

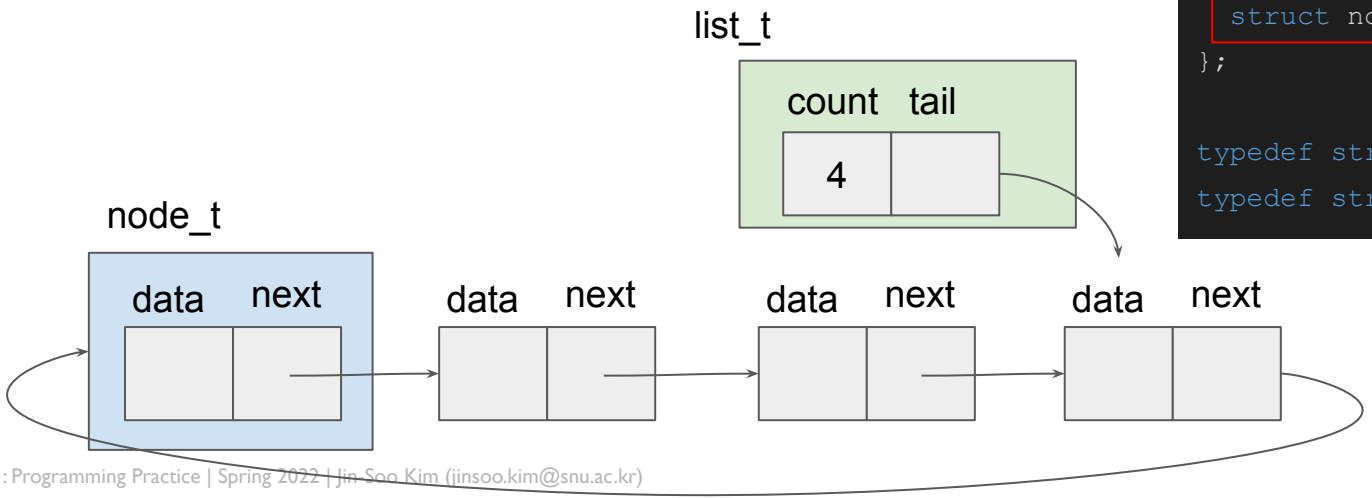
- Circular linked list
  - 마지막 노드가 첫번째 노드를 가리키는 linked list
  - 하나의 노드에서 링크를 계속 따라가다보면 모든 노드를 거쳐 자기 자신으로 돌아옴



실습 2 – 설명

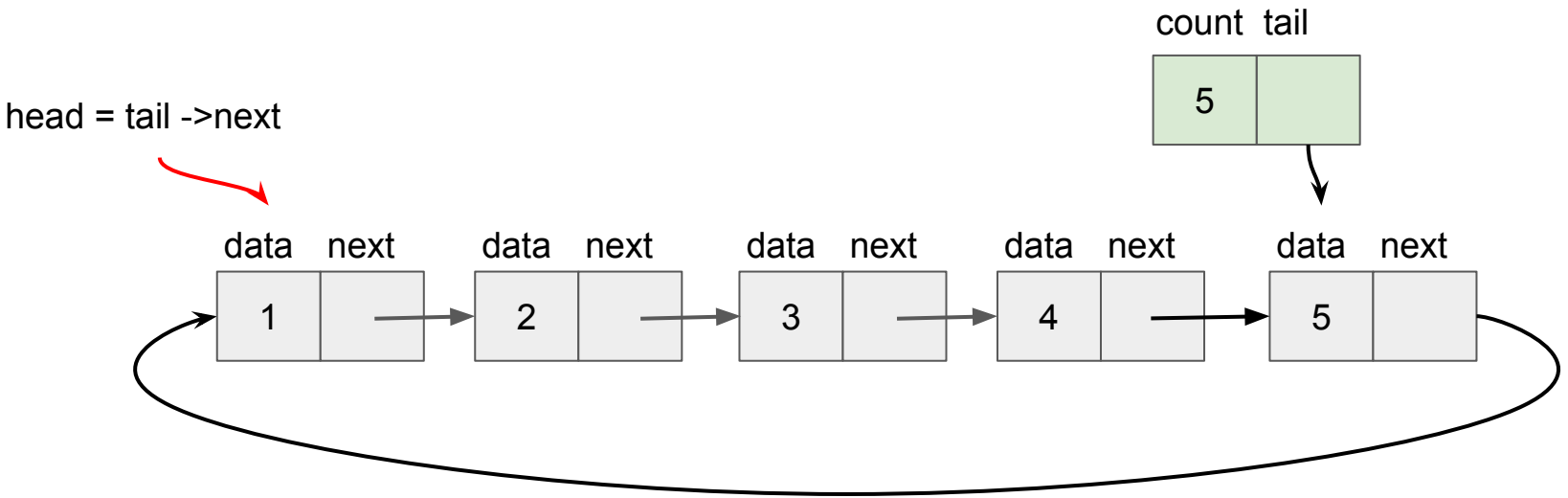
- Circular linked list를 위한 structure를 정의해봅니다.
- Singly linked list와는 다르게 마지막 노드를 가리키는 포인터 tail을 정의하여 사용해봅시다

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

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

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



실습 2 – print_list()

- List의 모든 노드를 출력하는 함수를 구현해 보세요
  - void print_list(list_t *list)
  - head부터 시작하여 자기자신으로 돌아올 때 까지 순회하며 출력



실습 2 – print_list()

- print_list()를 완성해보세요

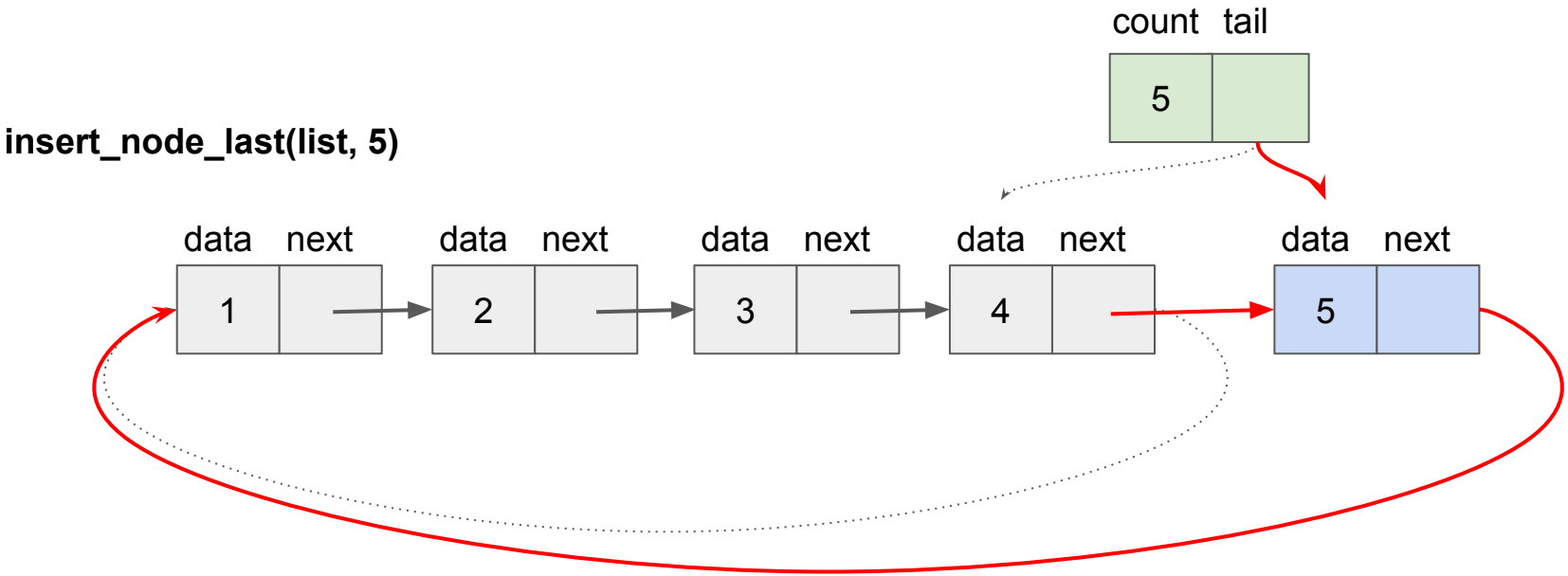
```
void print_list(list_t * list)
{
  printf("# elements: %d\n", list->count);
  printf("Data: ");

  if (/* write your code */) // 아무 노드도 없다면
    return;

  node_t * head = /* write your code */;
  node_t * ptr = head;

  /* write your code
     head node부터 시작하여,
     head로 돌아올때까지 순회하며 출력
     do while 문을 사용하면 쉽게 구현 가능.
  */

  printf("\n");
}
```

실습 2 – print_list()

```c
void print_list(list_t * list)
{
  printf("# elements: %d\n", list->count);
  printf("Data: ");

  if (list->tail == NULL)
    return;

  node_t * head = list->tail->next;
  node_t * ptr = head;

  do {
    printf("%d ", ptr->data);
    ptr = ptr->next;
  } while (ptr != head);

  printf("\n");
}
```

실습 2 – insert_node_last()

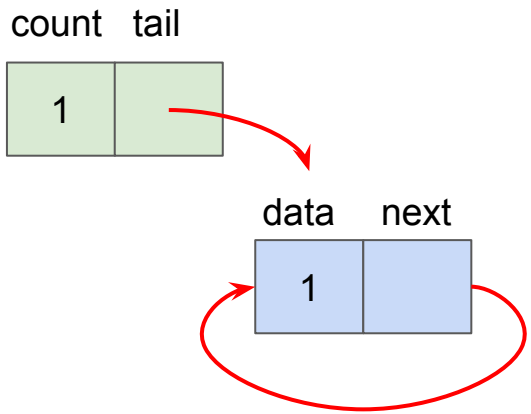
- List의 가장 뒤쪽에 node를 삽입하는 함수를 함께 구현해 봅시다
  - int insert_node_last(list_t *list, int data)



실습 2 – insert_node_last()

- List에 아무 노드도 존재하지 않을 경우 (tail이 NULL을 가리킬때)
  - 새로운 노드를 생성 한 후 자기 자신을 가리키게 함
  - List의 tail을 해당 노드를 가리키게 변경

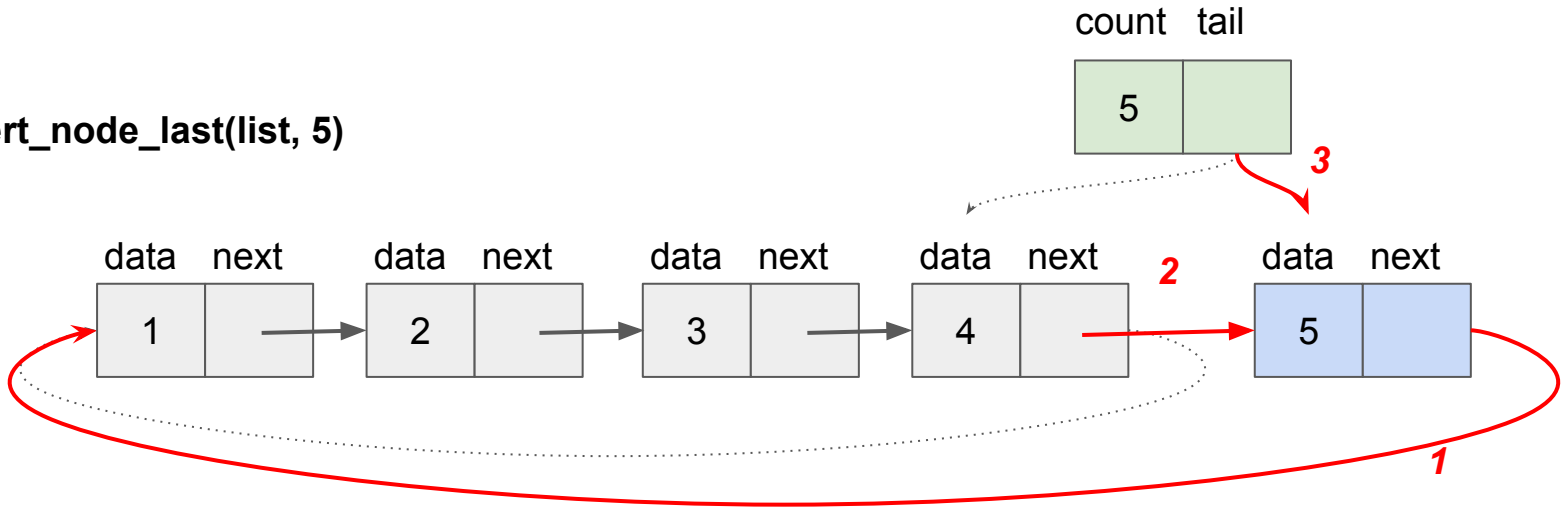
insert_node_last(list, 1)



실습 2 – insert_node_last()

- List에 노드가 존재할 경우
  - 1. 생성된 노드의 next 를 첫번째 노드로 연결 (tail 노드의 next를 가리키게 함)
  - 2. tail 노드의 next 를 현재 생성된 노드로 변경
  - 3. tail이 가리키는 노드를 현재 생성된 노드로 변경

insert_node_last(list, 5)

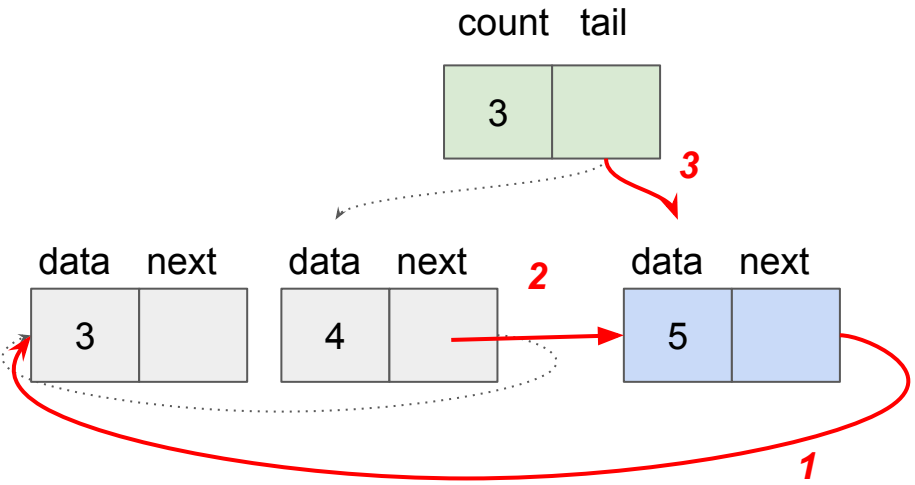


실습 2 – insert_node_last()

- inser_node_last()를 완성해보세요

```
int insert_node_last(list_t * list, int data) {
  node_t * new_node = malloc(sizeof(struct node));
  new_node->data = data;

  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  /* write your code
     tail이 NULL일때와 아닐때를 구분하여 노드 삽입
     list의 count 증가
  */

  printf("Insert node first %d\n", data);
  return 1;
}
```

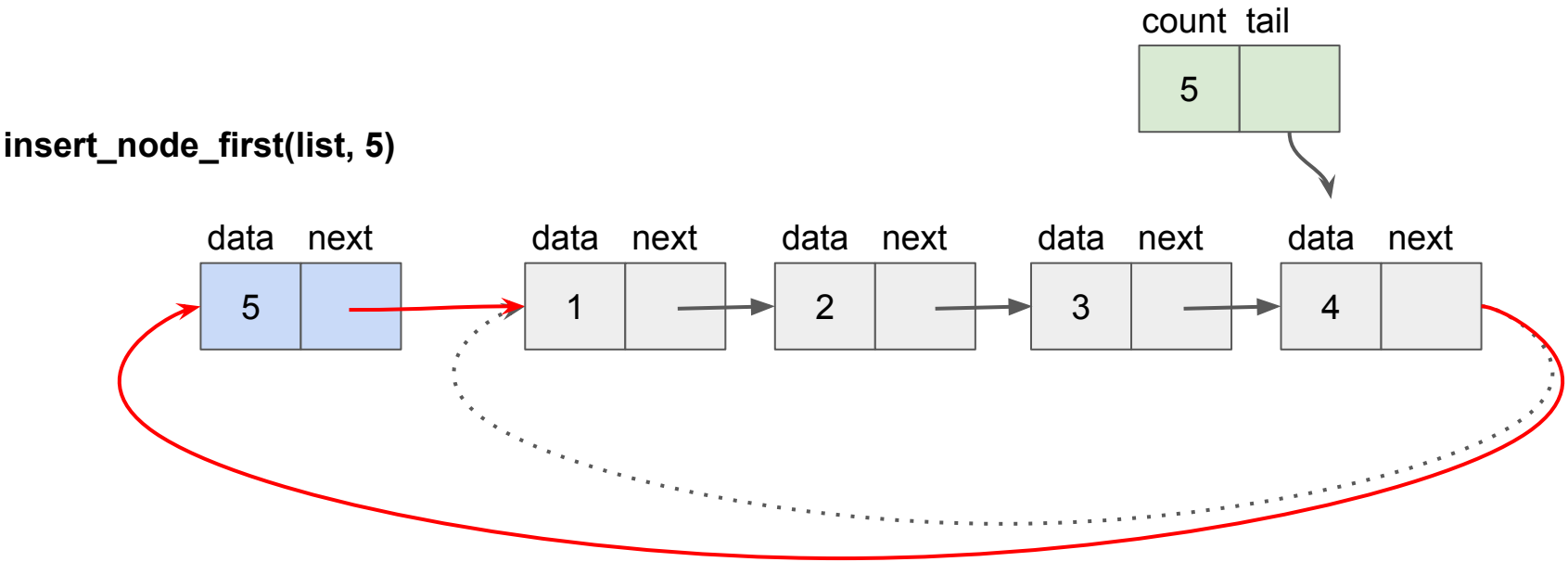
실습 2 – insert_node_last()




```c
int insert_node_last(list_t * list, int data) {
  node_t * new_node = malloc(sizeof(struct node));
  new_node->data = data;
  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  if (list->tail == NULL) {
    list->tail = new_node;
    new_node->next = new_node;
  }
  else {
  1 new_node->next = list->tail->next;
  2 list->tail->next = new_node;
  3 list->tail = new_node;
  }

  list->count++;
  printf("Insert node last %d\n", data);
  return 1;
}
```

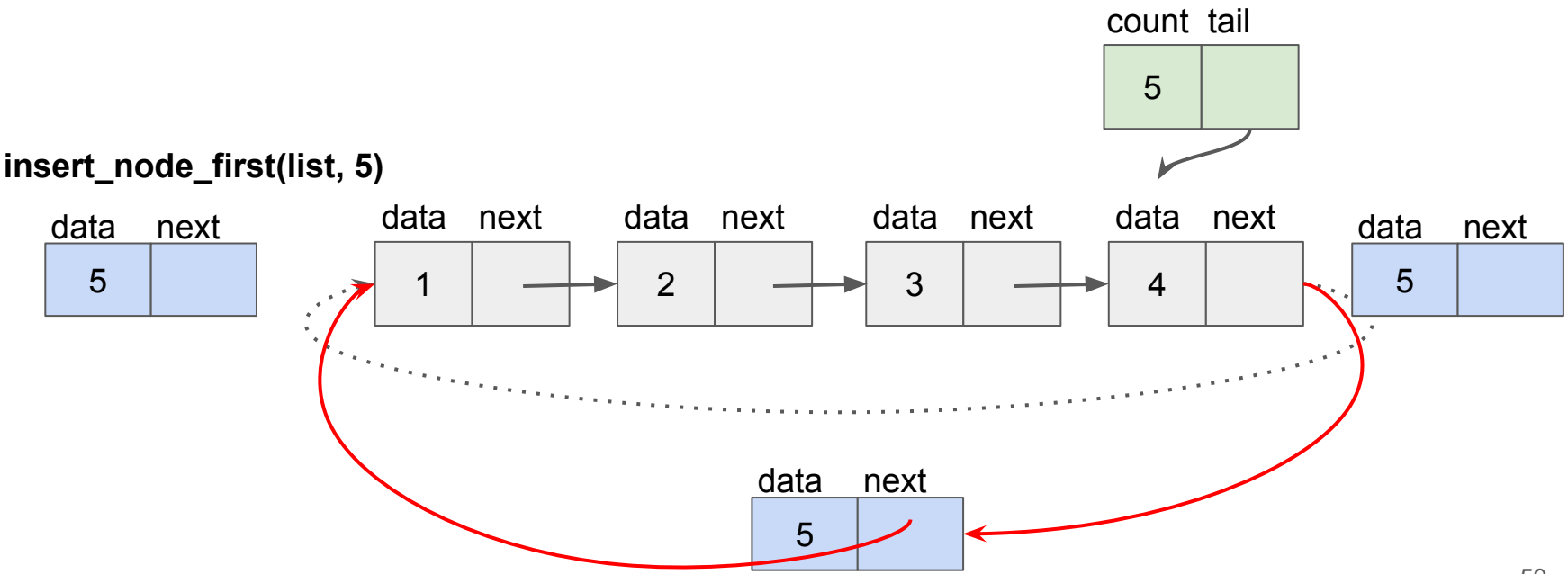
실습 2 – insert_node_first()

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



실습 2 – insert_node_first()

- tail이 변경되지 않는 것을 제외하고 insert_node_last()와 sequence가 동일합니다



실습 2 – insert_node_first()

- insert_node_first()를 완성해보세요

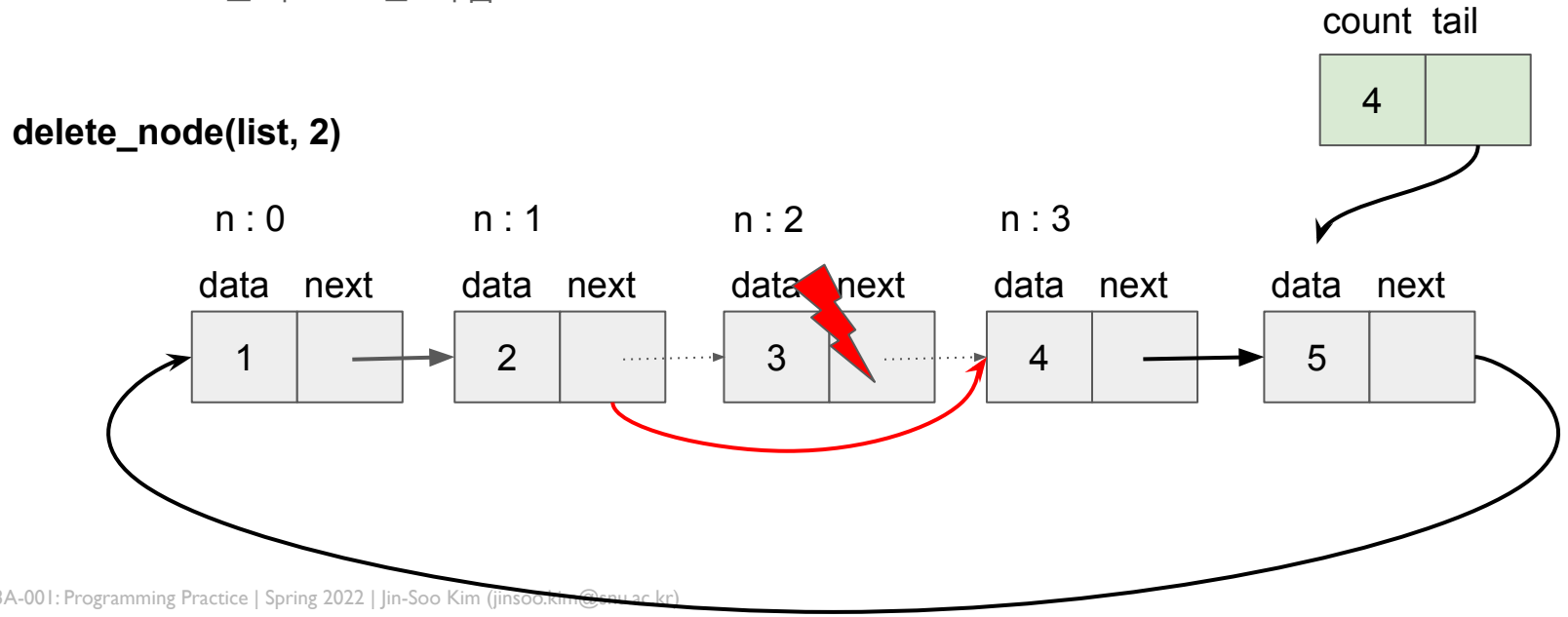
```
int insert_node_first(list_t * list, int data) {
  node_t * new_node = malloc(sizeof(struct node));
  new_node->data = data;

  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  /* write your code
     tail이 NULL일때와 아닐때를 구분하여 노드 삽입
     list의 count 증가
  */

  printf("Insert node first %d\n", data);
  return 1;
}
```

실습 2 – insert_node_first()

```c
int insert_node_first(list_t * list, int data) {
  node_t * new_node = malloc(sizeof(struct node));
  new_node->data = data;
  if (new_node == NULL){
    printf("Failed to create a node\n");
    return 0;
  }

  if (list->tail == NULL) {
    list->tail = new_node;
    new_node->next = new_node;
  }
  else {
    new_node->next = list->tail->next;
    list->tail->next = new_node;
    //list->tail = new_node;
    // inser_node_last()와 다르게 tail update를 하지 않음
  }
  list->count++;
  printf("Insert node last %d\n", data);
  return 1;
}
```

실습 2 – delete_node() – 제출

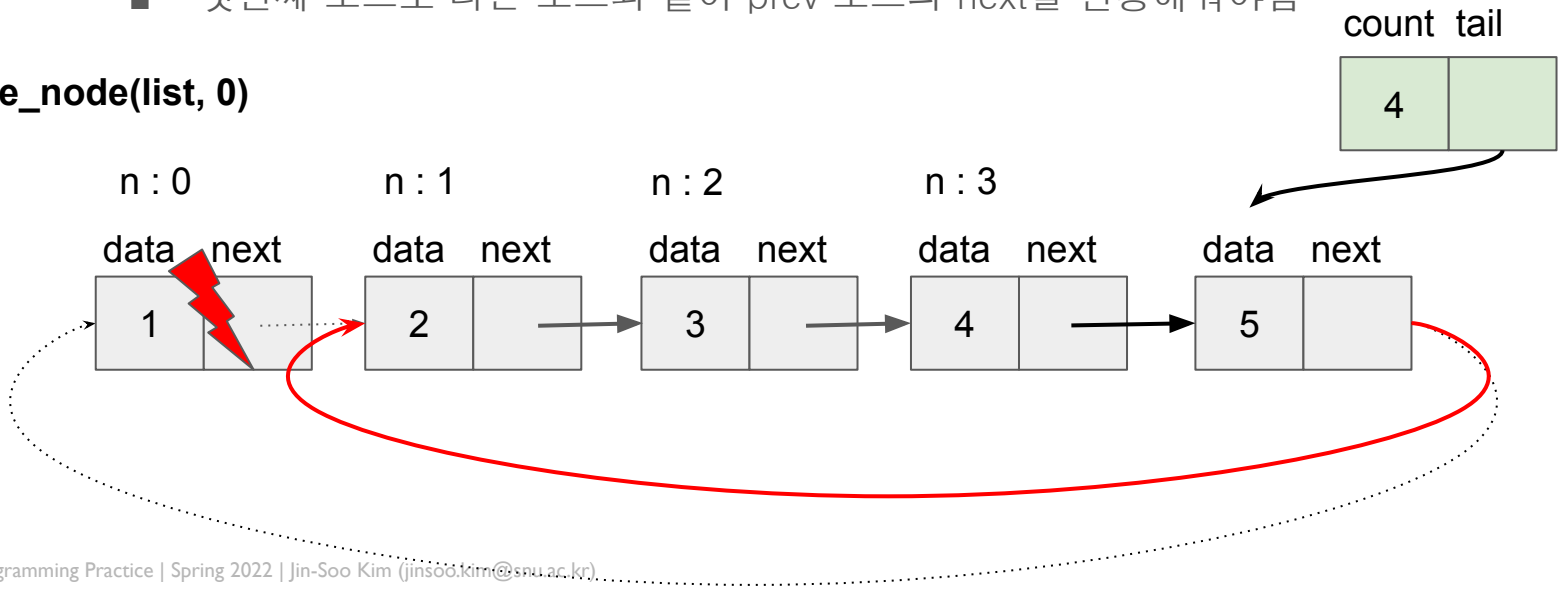
- List 중간에 node를 제거하는 함수를 구현해보세요
  - int delete_node(list_t *list, int n)
  - n번째 노드를 지움



실습 2 – delete_node()

- Singly linked list와 다른점은 제일 첫번째 노드를 제거할 경우입니다
  - Singly linked list에서 첫번째 노드를 제거할 경우, head만 옮겨주면 되었지만 circular linked list에서는 가장 마지막 노드(tail)의 next를 첫번째 노드의 next로 설정해주어야함
    - 첫번째 노드도 다른 노드와 같이 prev 노드의 next를 변경해줘야함

delete_node(list, 0)



실습 2 – delete_node()

- delete_node()를 완성해보세요

```
int delete_node(list_t * list, int n) {

  if (list->tail == NULL)
    return -1;

  node_t * head = /* write your code */ ;
  node_t * prev = /* write your code */ ;
  node_t * cur = head;

  /*
    cur와 prev를 한 노드씩 움직이면서 pos번째 node를 찾음
    prev의 next를 cur의 next로 바꿔주어 cur를 지움
    현재 tail노드가 지워졌을 경우, tail을 prev로 변경
    지우고 난 후 리스트에 아무것도 없다면 tail을 NULL로 변경
    tail node가 변경되었다면 tail update
  */
  return /* write your code 지운 노드의 data를 return */ ;
}
```

실습 2 – delete_node()

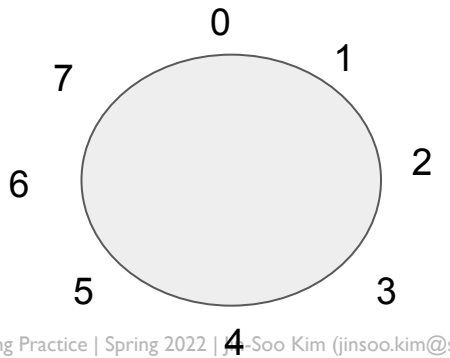
```c
int delete_node(list_t * list, int n) {

  if (list->tail == NULL)
    return -1;

  node_t * head = list->tail->next;
  node_t * prev = list->tail;
  node_t * cur = head;
  int data;

  // cur를 n번째 노드를 가리키게 반복문을 수행
  for (int i = 0; i < n; i++) {
    prev = cur;
    cur = cur->next;
  }
```

```c
  if (cur->next == cur) { // # elements == 1
    list->tail = NULL;
  }
  else {
    prev->next = cur->next;
    // tail 노드가 지워졌다면
    if (cur == list->tail)
        list->tail = prev;
  }

  list->count--;
  data = cur->data;
  free(cur);

  printf("Delete node %d\n", data);
  return data;
}
```

실습 3

실습 3 – 제출

- 요세푸스 순열을 구하는 프로그램을 만들어보세요
  - 0 ~ N-1번까지 N명의 사람이 원을 이루며 앉아있음
  - 순서대로 K번째 사람을 제거하고, 한사람이 제거되면 남은 사람들로 이루어진 원을 따라 이 과정을 계속 해감 (K > 0)
  - N명의 사람이 모두 제거될때까지 계속됨
  - 원에서 사람들이 제거되는 순서를 (N,K) – 요세푸스 순열이라고 함



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

실습 3 – Tip

- 실습 2에서 작성했던 Circular linked list를 사용해보세요
- head에서부터 K번째 노드를 제거하고 tail을 K – 1번째 노드를 (head는 K + 1번째) 가리키게 변경하면서 모든 노드가 제거될때까지 반복해보세요

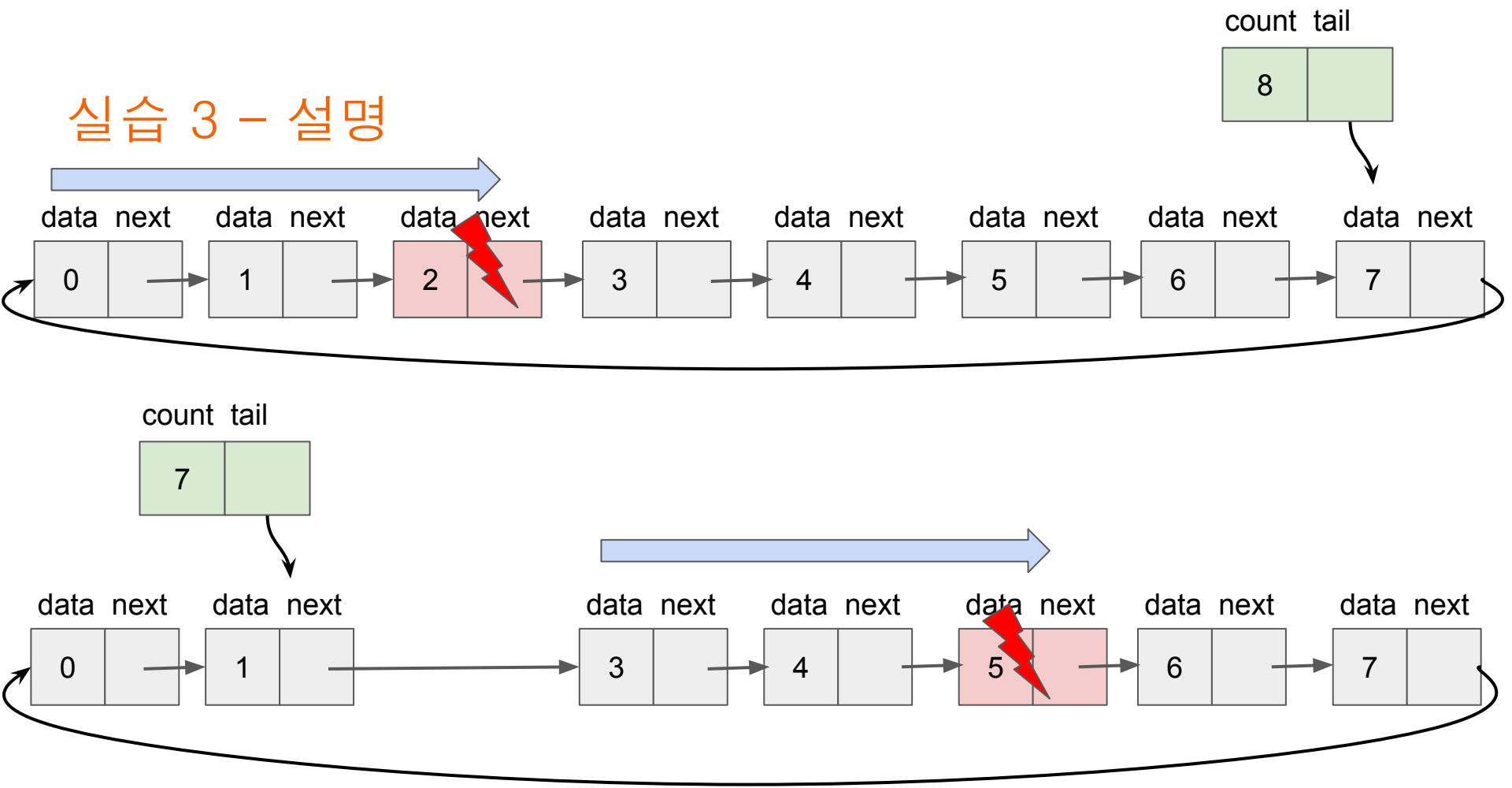
실습 3

- 기존에 구현했던 insert_node_last()와 delete_node()함수를 사용해주세요

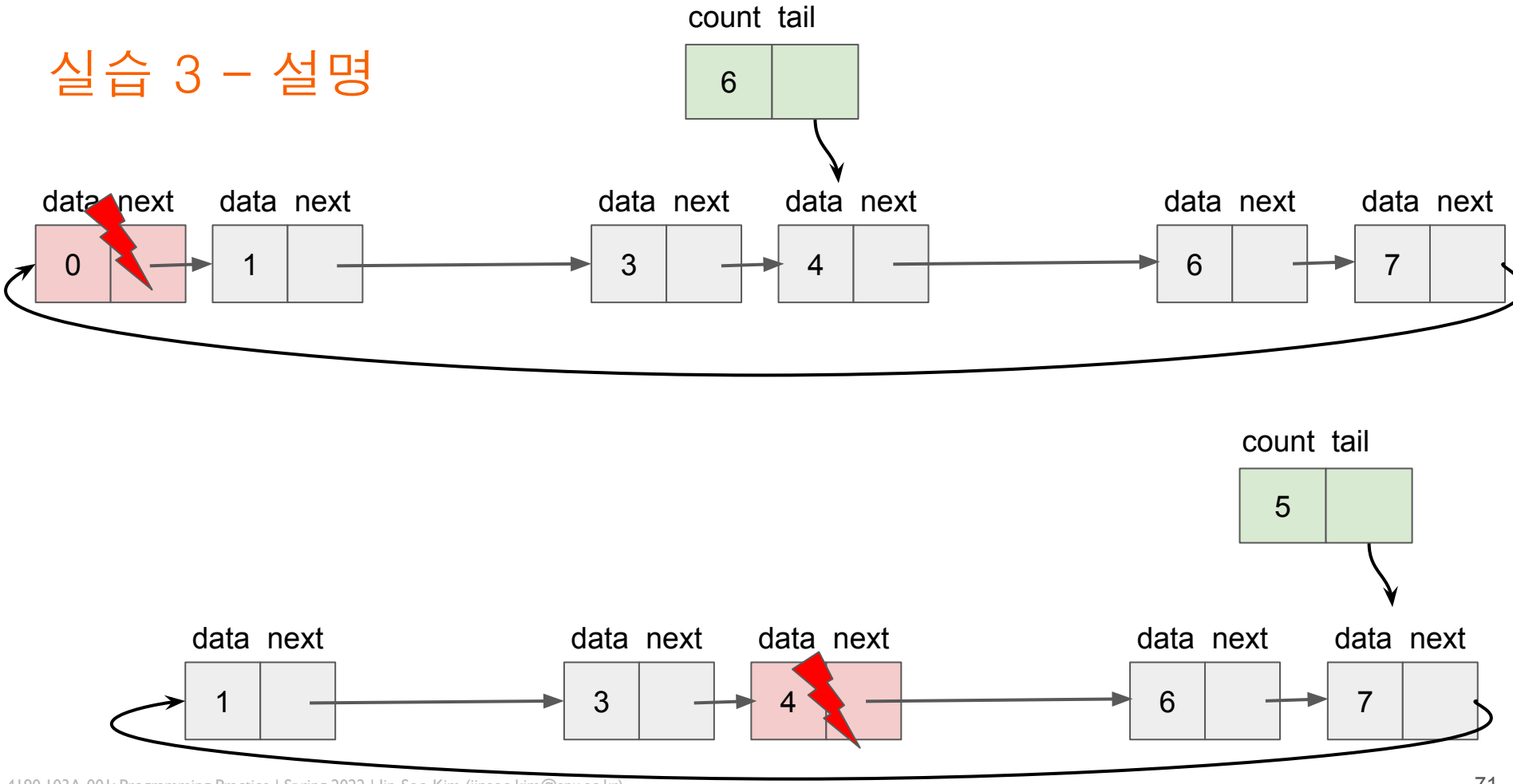
```
make -s
./main
7 2
2 5 1 6 4 0 3
./main
8 3
3 7 4 1 0 2 6 5
```

```c
int main(void) {
  list_t list;
  list.tail = NULL;
  list.count = 0;
  int n,k,d;
  scanf("%d %d", &n, &k);

  /*
    Step #1 : 0부터 n-1까지 list 마지막부분에
              순차적으로 삽입함
  */

  /*
    Step #2: list에 아무노드도 존재하지 않을때까지
             규칙에 따라서 노드를 제거함
             노드 제거할때마다 tail 노드 변경 필요
  */

  printf("\n");
  return 0;
}
```

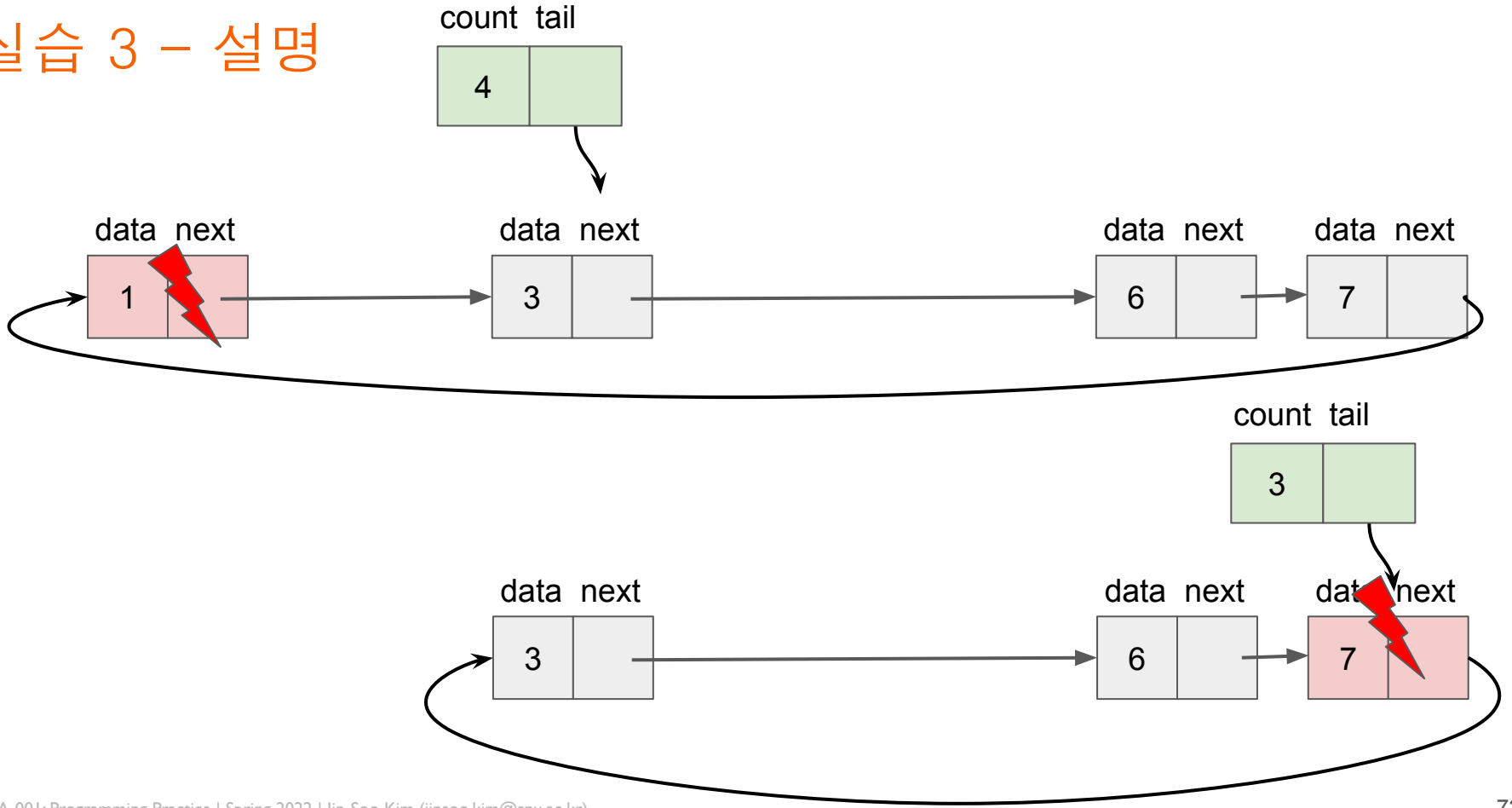
실습 3 – 설명


count tail
8
data next
0
1
2
3
4
5
6
7
count tail
7
data next
0
1
3
4
5
6
7

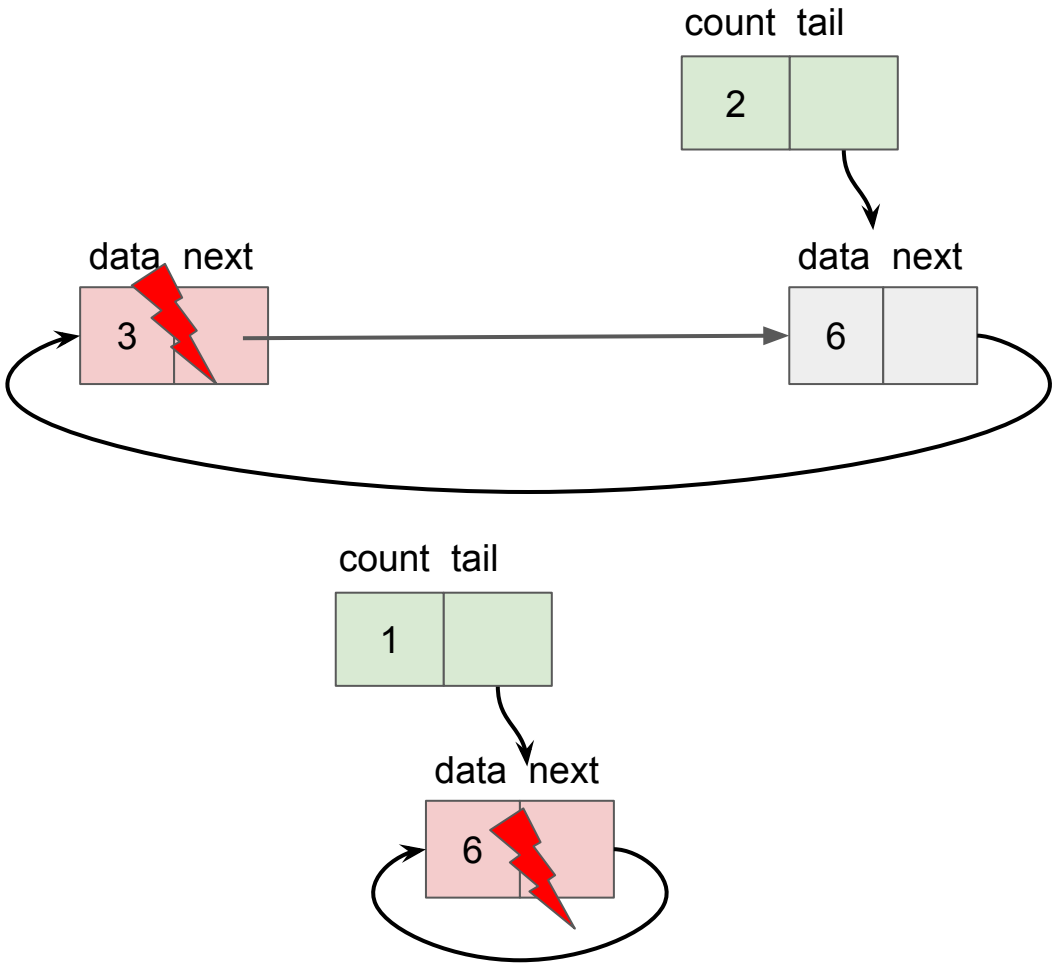
실습 3 – 설명


count tail
6
data next
0
data next
1
data next
3
data next
4
data next
6
data next
7
count tail
5
data next
1
data next
3
data next
4
data next
6
data next
7

실습 3 – 설명


count tail
4
data next
1
data next
3
data next
6
data next
7
count tail
3
data next
3
data next
6
dat next
7

실습 3 – 설명



실습 3 – 풀이

```c
node_t * search_node(list_t * list, int n)
{
  if (list->tail == NULL)
    return NULL;

  node_t *head = list->tail->next;
  node_t *cur = head;
  for (int i = 0; i < n; i++)
    cur = cur -> next;

  return cur;
}
```

```c
int main(void) {
  list_t list;
  list.tail = NULL;
  list.count = 0;
  node_t * prev;
  int n,k,d;
  scanf("%d %d", &n, &k);

  for (int i = 0; i < n; i++)
     insert_node_last(&list, i);

  while(1) {
    prev = search_node(&list, k-1);
    d = delete_node(&list, k);
    printf("%d ",d);

    if (list.tail == NULL) break;
    else list.tail = prev;
  }
  printf("\n");
  return 0;
}
```

텀 프로젝트 공지

텀 프로젝트 공지

- Wordle
  - Wordle - The New York Times (nytimes.com)
- Due date : 6월 18일 23시 59분
- 텀 프로젝트 OT는 다음주 실습시간(5/25)에 진행할 예정입니다.

텀 프로젝트 팀 편성

- 팀 별 인원수 : 3명
- 자유롭게 팀을 구성해주세요.
- 아래 구글 폼에 조 이름과 인원 정보를 입력해주세요.
  - 인원수가 2명이라도 구글 폼에 작성 해주시길 바랍니다.
  - 나머지 인원은 random으로 배정합니다.
- 다음주 월요일 자정까지 조를 못 이룬 학생들은 random으로 팀 편성하겠습니다.
- ETL에 팀 모집을 위한 게시판을 만들었으니 적극 활용해주시길 바랍니다.
- https://docs.google.com/forms/d/17PCWPwrj4981TYdqJeYQr5cd5PPAhQ0tnywg6PNNlL0/edit