

# Data structure Analysis

## Project #1: MyLib

CSE4100: System Programming

Sungyong Park and Youngjae Kim (PhD)

Data Intensive Computing and Systems Laboratory (DISCOS)

<https://discos.sogang.ac.kr>

TA: Kyuri Park and Joseph Ro

# 목차

## ■ 프로젝트 개요

## ■ 배경지식

1. List
2. Hash table
3. Bitmap

## ■ 자료구조 함수 분석

# 프로젝트 개요

“일반적인 커널 코드에서 쓰이는 세개의 자료구조 분석 및 구현”

## 1. Data Structures in MyLib

- List, Hash Table, Bitmap 세개의 자료구조를 분석 및 추가 구현

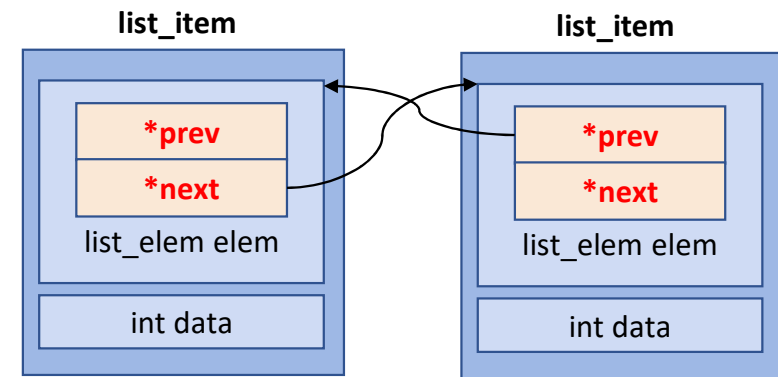
## 2. Interactive Program

- 명령어를 입력 받아 구현한 자료구조의 functionality를 확인할 수 있는 프로그램 개발

※ 해당 PPT에서 설명하는 내용은 MyLib에서 사용되는 자료구조에 대한 배경지식 설명입니다.  
프로젝트 진행과 관련한 자세한 내용은 프로젝트 명세서를 확인 바랍니다.

# 배경지식 #1: List

- MyLib에서 사용되는 리스트의 종류는 “**doubly linked list**” 이다.
- 이는 일반적인 리스트 구조와 다르기 때문에, PPT에서 설명하는 구조를 이해하시고 코드를 짜는 것을 추천함
- Mylib의 list entry (list\_item)는 pointer (list\_elem)와 데이터를 나누어서 저장함  
(Described in next slide)



Split the pointer and data

✗

```
struct list_elem
{
    struct list_elem *prev;
    struct list_elem *next;
}

struct list_item
{
    struct list_elem elem;
    int data;
    /* Other members you want */
}
```

✗ 'struct list\_item' is not given in the source code, you need to implement it.

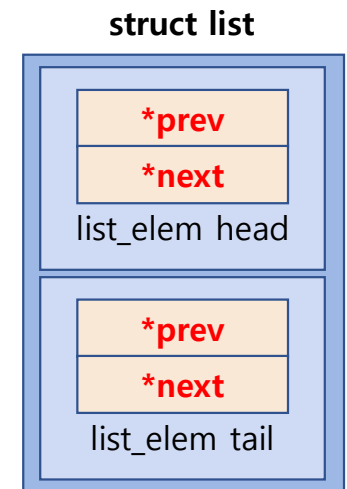
# 배경지식 #1: List

## ■ struct list\_elem

- 리스트와 관련된 데이터를 저장하는 자료구조는 list\_elem 자료구조를 포함하고 있어야 함.
- List를 다루는 모든 함수들은 list\_item 자료구조가 아닌 list\_elem 자료구조를 통해 수행됨.
- 현재 skeleton code에선 list\_elem 자료구조만 구현되어 있음.
- 따라서, list\_elem과 data를 포함하는 list\_item 자료구조를 추가 구현하는 것이 필요함.

## ■ Struct list

- List 자료구조는 list의 head와 tail을 저장하고 있음.  
이 head와 tail 역시 list\_elem 자료구조로 이루어짐.



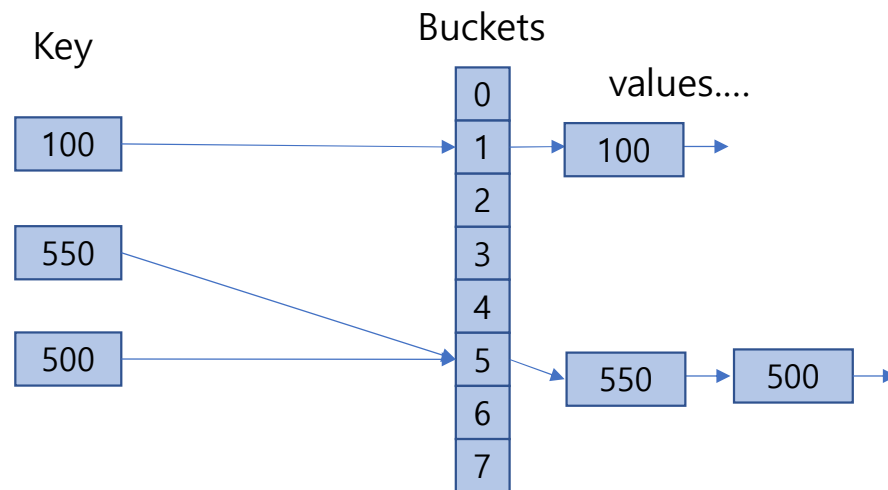
※ 'struct list\_item' is not given in the source code, you need to implement it.

# 배경지식 #2: Hash Table

- A hash table은 **keys** 와 **values** 로 연관된 자료구조이다.
- The primary operation is a lookup.
  - Key가 주어지면, key에 해당하는 value를 찾는 것
- 특정 **hash function**을 이용하여 key를 hash로 바꿔주는 역할을 함

※ Assume that key and value are same.

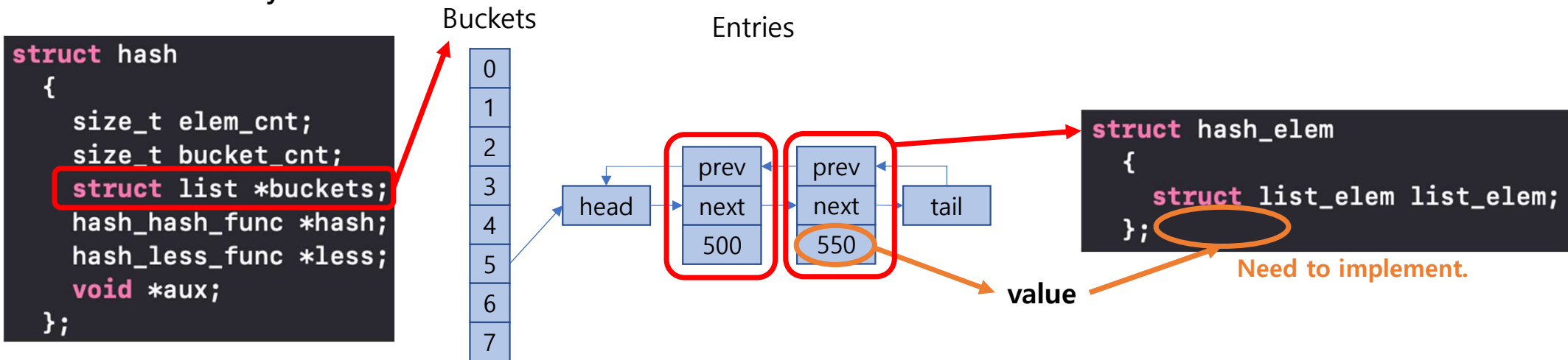
$$\text{hash(key)} = \text{key} / 100 \\ = \text{bucket ID}$$



# 배경지식 #2: Hash Table

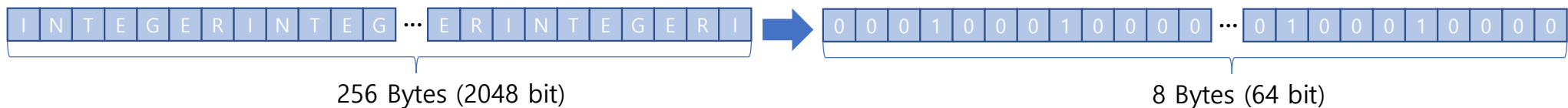
- 각 bucket은 **list structure**로 구성되어 있음(we covered in previous slides.)
- 각 value들은 entry로 구성되어 있음.
  - 이 때, entry는 **hash\_elem structure**로 구성
- In hash\_elem, splits list element pointers and data. (same as list)

※ Assume that key and value are same.



# 배경지식 #3: Bitmap

- A bit array(or bitmap, in some cases) 는 각각의 Boolean value로 저장되는 array이다.
- A bitmap은 memory space 낭비를 줄일 수 있음 (아래 그림 참고)



- MyLib에서 사용되는 bitmap 자료구조는 usual bitmap이랑 다르게 구성되어 있음
  - 기능은 동일
- Bitmap in MyLib

```
typedef unsigned long elem_type;
struct bitmap
{
    size_t bit_cnt;
    elem_type *bits;
};
```



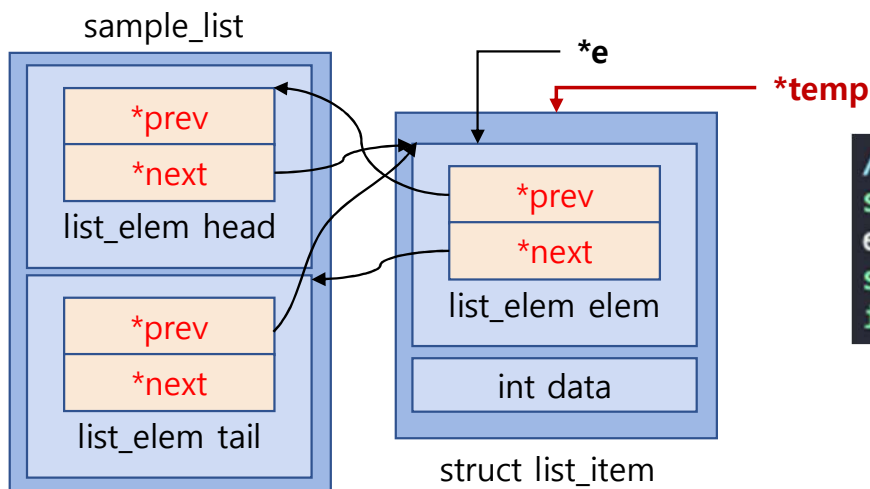
# 자료구조 분석 #1: List Function in MyLib

- **void list\_init(struct list \*list)**
  - Initializes LIST as an empty list.
  - It should be executed before an element is inserted in LIST.
- **struct list\_elem\* list\_begin(struct list \*list)**
  - Returns the first element of LIST.
  - Usually used to iterate the LIST.
- **struct list\_elem\* list\_next(struct list\_elem \*elem)**
  - Returns the next element of ELEM.
  - Usually used to iterate the LIST or search ELEM in the LIST.
- **struct list\_elem\* list\_end(struct list \*list)**
  - Returns the last ELEM in the LIST.
  - Usually used to iterate the LIST.

# 자료구조 분석 #1: List Function in MyLib

## ■ #define list\_entry(list\_elem, struct, member)

- Converts the pointer from list\_elem into a pointer of list\_item that list\_elem is embedded.
- Usually used to get address of list\_item which embeds list\_elem.



```
/* Assume that there already exists a list, sample_list */
struct list_elem *e;
e = list_begin (&sample_list);
struct list_item *temp = list_entry(e, struct list_item, elem);
int temp_data = temp->data;
```

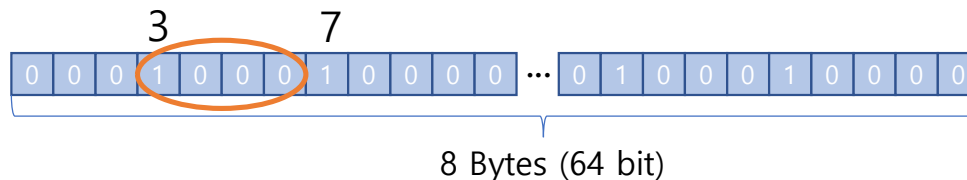
By using list\_entry, we can get address of list\_item

# 자료구조 분석 #2: Hash Table Function in MyLib

- **void hash\_init(struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux)**
  - Initializes hash table H and sets hash function HASH and comparison function LESS.
  - You can see the example of hash function such as hash\_int, hash\_bytes, and hash\_string.  
(You have to use hash\_int function to pass the test.)
  - Comparison function LESS is used to compare two hash elements.
- **void hash\_apply(struct hash \*h, hash\_action\_func \*action)**
  - You can apply any ACTION function which you made to hash table H.
  - Used for applying specific function to all elements in hash table.  
e.g.) square function.
  - You can learn the usage of it from 'hash\_apply.in' and 'hash\_apply.out' in tester directory.
- **#define list\_elem\_to\_hash\_elem(LIST\_ELEM) list\_entry(LIST\_ELEM, struct hash\_elem, list\_elem)**
  - Converts pointer to LIST\_ELEM into a pointer HASH\_ELEM

## 자료구조 분석 #3: Bitmap Function in MyLib

- **struct bitmap \*bitmap\_create(size\_t bit\_cnt)**
  - Initializes a bitmap of BIT\_CNT bits and sets all its bits to false.
- **void bitmap\_set (struct bitmap \*b, size\_t idx, bool value)**
  - Atomically sets the bit numbered IDX in B to VALUE.
- **size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value)**
  - Returns the number of bits in B between START and START + CNT, exclusive, that are set to VALUE.



```
bitmap_count(b, 3, 4, 1) == 1
```