Project #1 : MyLib

담당 교수 : 김영재

학번 : 20211558

이름 : 윤준서

I. Additional Implementation

Data	`struct list list_data`
Function	생성한 리스트와 해시테이블 데이터를 저장하는 전체 데이터 리스트.

Data	`struct bitmap *list_bitmap[BIT_MAX] = {0,}`
Function	비트맵 데이터를 저장하는 데이터 리스트.

Prototype	`void cmd_par (char cmd[CMD_LEN], char arg[][ARG_LEN], int *arc)`
Parameter	stdin 입력 문자열, 입력 문자열 저장 리스트, 키워드 개수
Return	void
Function	입력 문자열 cmd를 공백(' ')을 기준으로 분리하여 arg에 저장한다. 그리고 arg의 원소 개수를 arc를 통해 저장한다.

Prototype	`int toInt(char arg[ARG_LEN])`
Parameter	stdin에서 입력한 숫자 문자열
Return	int : 문자열을 정수로 변환한 값
Function	stdin에서 입력한 문자열 중 숫자를 정수 값으로 변환한다.

Prototype	`void instruction()`
Parameter	none
Return	void
Function	main()을 간단하게 만들기 위해 입력 명령어를 해당 함수에서 진행한다.

Prototype	`int main()`
Parameter	none
Return	0
Function	데이터 리스트를 초기화하고, instruction() 함수를 실행한다. 이때 데이터 리스트는 생성한 리스트와 해시테이블 데이터를 저장하는 전체 데이터 구조다.

II. List

Data	`union data { int val; struct list *list; struct hash *hash; struct bitmap *bitmap; };`
Function	자료형에 따른 데이터와 키 값을 저장 및 판단할 때 사용하는 공용체.

Data	`struct list { struct list_elem head; /* List head. */ struct list_elem tail; /* List tail. */ char name[NAME_LEN]; };`
Function	리스트 데이터에 이름 문자열 추가.

Data	`struct list_item { struct list_elem elem; int type; union data data; };`
Function	list_elem, 자료형, 해당 데이터 자체를 가지는 구조체.

Prototype	`struct list_elem *get_elem (struct list *, int idx)`
Parameter	찾고자 하는 원소가 속해 있는 리스트 데이터, 해당 원소의 인덱스
Return	struct list_elem * : 해당 원소의 list_elem 구조체 주소
Function	리스트의 특정 인덱스의 원소를 반환한다.

Prototype	`struct list_elem *find_list_elem (struct list *, char name[NAME_LEN])`
Parameter	데이터 리스트, 찾고자 하는 데이터의 이름 문자열
Return	struct list_elem * : 해당 데이터의 list_elem 구조체 주소
Function	생성한 리스트와 해시테이블 데이터를 저장하는 전체 데이터 구조인 데이터 리스트에서 이미 생성한 리스트를 이름을 찾아서 반환한다. 데이터 리스트에서 데이터 값을 출력 또는 삭제할 때, 데이터를 찾는 데 쓰인다.

Prototype	`struct list *find_list (struct list *, char name[NAME_LEN])`
Parameter	데이터 리스트, 찾고자 하는 리스트의 이름 문자열
Return	struct list * : 해당 데이터의 list 구조체 주소
Function	생성한 리스트와 해시테이블 데이터를 저장하는 전체 데이터 구조인 데이터 리스트에서 이미 생성한 리스트를 이름을 찾아서 반환한다. 이때, list_elem 구조체가 아닌 list 구조체를 반환한다. 리스트 함수를 실행할 때, 리스트를 찾는 데 쓰인다.

Prototype	`void list_swap (struct list_elem *a, struct list_elem *b)`
Parameter	리스트 내에서 위치를 바꾸려는 두 list_elem 구조체
Return	void
Function	리스트 데이터 내에서 두 원소의 위치를 서로 바꾼다.

Prototype	`void list_shuffle (struct list *list)`
Parameter	내부 순서를 바꾸려는 리스트 데이터
Return	void
Function	리스트 데이터 내의 모든 원소의 위치 순서를 섞는다.

Prototype	`void list_delete (struct list *)`
Parameter	데이터 리스트에서 삭제하려는 리스트 데이터
Return	void
Function	생성한 리스트 데이터와 해시테이블을 저장하는 전체 데이터 구조인 데이터 리스트에서 특정 리스트를 삭제한다.

III.Hash Table

Data	`struct hash { size_t elem_cnt; /* Number of elements in table. */ size_t bucket_cnt; /* Number of buckets, a power of 2. */ struct list *buckets; /* Array of `bucket_cnt' lists. */ hash_hash_func *hash; /* Hash function. */ hash_less_func *less; /* Comparison function. */ void *aux; /* Auxiliary data for `hash' and `less'. */ char name[NAME_LEN]; };`
Function	해시 데이터 구조체에 이름 문자열 추가.

Prototype	`bool hash_less(const struct hash_elem *a, const struct hash_elem *b, void *aux)`
Parameter	해시 데이터 값 크기를 비교하려는 두 hash_elem 구조체, 보조 포인터
Return	bool
Function	두 해시 데이터의 키 값을 비교한다. b의 값이 더 크면 true, a의 값이 더 크면 false를 반환한다.

Prototype	`void hash_action(struct hash_elem *elem, void *aux)`
Parameter	데이터 리스트에서 삭제하려는 해시테이블, 보조 포인터
Return	void
Function	생성한 리스트 데이터와 해시테이블을 저장하는 전체 데이터 구조인 데이터 리스트에서 특정 해시테이블 삭제한다.

Prototype	`unsigned hash_hash_int(const struct hash_elem *e, void *aux)`
Parameter	해시 값을 찾고자 하는 해시 데이터, 보조 포인터
Return	unsigned int : 해시 값
Function	해시 데이터의 해시 값을 hash_int()를 통해 반환하기 위해 hash_int()를 실행한다. 그리고 얻은 해시 값을 반환한다.

Prototype	`struct hash *find_hash(struct list *list, char name[NAME_LEN])`
Parameter	데이터 리스트, 찾고자 하는 해시테이블의 이름 문자열
Return	struct hash * : 해당 데이터의 hash 구조체 주소
Function	생성한 리스트 데이터와 해시테이블을 저장하는 전체 데이터 구조인 데이터 리스트에서 이미 생성한 해시테이블을 이름을 찾아서 반환한다.

Prototype	`unsigned int hash_int_2 (int num)`
Parameter	해시 값을 만들고자 하는 정수
Return	unsigned int : 해시 값
Function	입력한 정수를 새로운 해시 값으로 만드는 두 번째 해시함수. DJB2 알고리즘을 정수형으로 변형했다.

Prototype	`void hash_squared(struct hash_elem *elem, void *aux)`
Parameter	해시 데이터, 보조 포인터
Return	void
Function	해시 데이터의 키 값을 제곱해서 다시 저장한다. hash_apply() 함수를 실행할 때 사용한다.

Prototype	`void hash_cubed(struct hash_elem *elem, void *aux)`
Parameter	해시 데이터, 보조 포인터
Return	void
Function	해시 데이터의 키 값을 세제곱해서 다시 저장한다. hash_apply() 함수를 실행할 때 사용한다.

IV. Bitmap

Prototype	`void bitmap_expand(struct bitmap *, int size)`
Parameter	비트맵 데이터, 크기 정수
Return	void
Function	비트맵 데이터의 크기를 size 만큼 확장한다.