**Pintos Project 0-2: Pintos Data Structure**

|  |  |
| --- | --- |
| 담당 교수 : | 문의현 |
| 학번 : | 20180492 |
| 이름 : | 김범준 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem\*b){} |
| **Parameter** | list\_elem 포인터 a와 b. 리스트 내 원소가 바뀔 구조체의 포인터. |
| **Return** | 없음 |
| **Function** | 함수 내부에서 list\_entry 매크로를 이용하여 인자로 받은 두 원소의 상위 구조체인 list\_item에 접근한다. 그 후 list\_item의 멤버 변수인 정수형 데이터를 서로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list\* list){} |
| **Parameter** | 원소를 무작위로 배열할 리스트의 포인터. |
| **Return** | 없음. |
| **Function** | 리스트의 크기 만큼 반복문이 돌아가면서 0~(크기 – 1) 범위의 난수를 생성하여 반복문의 인덱스(i)의 위치에 해당하는 원소와 난수의 위치에 해당하는 원소의 위치를 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size){} |
| **Parameter** | 크기를 확장할 비트맵의 포인터와 확장할 크기의 정수 |
| **Return** | 확장된 비트맵의 구조체 포인터 |
| **Function** | 인자로 들어온 비트맵의 크기 + size의 크기를 가지는 새로운 비트맵을 생성한 뒤 기존 비트맵의 정보를 복사하고, 확장된 비트는 0의 값을 그대로 둔다. 그 후 새로운 비트맵을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i){} |
| **Parameter** | 해시테이블에 저장된 정수 데이터. |
| **Return** | I mod 1,000,000,007 |
| **Function** | 10억을 넘는 가장 작은 소수로 나눈 나머지를 반환하는 사용자 지정 해시 함수. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux){} |
| **Parameter** | list\_elem 구조체 포인터 a와 b, void형 포인터 aux |
| **Return** | a의 데이터가 b보다 작으면 true, 이외에는 false |
| **Function** | list\_entry 매크로를 이용해 a와 b의 데이터 값에 접근하여 둘의 대소를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_hash(const struct hash\_elem \*a, void \*aux) |
| **Parameter** | 해시에 저장되는 데이터(hash\_elem\*) a, void형 포인터 aux |
| **Return** | 데이터를 hash\_int함수에 인자로 넣은 unsigned형 반환값 |
| **Function** | hash\_entry를 통해 a의 데이터에 접근하여 해시 함수를 통해 해시키를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux){} |
| **Parameter** | hash\_elem 구조체 포인터 a와 b, void형 포인터 aux |
| **Return** | a의 데이터가 b보다 작으면 true, 이외에는 false |
| **Function** | hash\_item 매크로를 이용해 a와 b의 데이터 값에 접근하여 둘의 대소를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destructor(struct hash\_elem \*a, void \*aux){} |
| **Parameter** | 메모리 할당 해제할 hash\_elem 구조체 포인터 a, void형 포인터 aux |
| **Return** | 없음 |
| **Function** | hash\_elem의 상위 구조체에 hash\_entry 매크로를 이용하여 접근한 뒤, 메모리 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem \*a, void \*aux){} |
| **Parameter** | 데이터를 제곱할 hash\_elem 구조체 포인터 a, void형 포인터 aux |
| **Return** | 없음 |
| **Function** | hash\_entry 매크로를 이용하여 a의 데이터에 접근한 뒤, 해당 데이터를 제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem \*a, void \*aux){} |
| **Parameter** | 데이터를 세제곱할 hash\_elem 구조체 포인터 a, void형 포인터 aux |
| **Return** | 없음 |
| **Function** | hash\_entry 매크로를 이용하여 a의 데이터에 접근한 뒤, 해당 데이터를 세제곱한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head(struct list\_elem \*elem){} |
| **Parameter** | list\_elem 구조체 포인터 |
| **Return** | 해당 원소가 리스트의 head면 true, head가 아니면 false |
| **Function** | 해당 원소가 리스트의 head면 true, head가 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior(struct list\_elem \*elem){} |
| **Parameter** | list\_elem 구조체 포인터 |
| **Return** | 해당 원소가 리스트의 head이거나 tail이면 false, 이외에는 true |
| **Function** | 해당 원소가 리스트의 head이거나 tail이면 false, 이외에는 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail(struct list\_elem \*elem){} |
| **Parameter** | list\_elem 구조체 포인터 |
| **Return** | 해당 원소가 리스트의 tail이면 true, 아니면 false |
| **Function** | 해당 원소가 리스트의 tail이면 true, 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 없음. |
| **Function** | 빈 리스트 하나를 생성한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트 맨 앞 원소의 list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 가장 앞 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next(struct list\_elem \*elem){} |
| **Parameter** | list\_elem 구조체 포인터 |
| **Return** | 인자로 받은 원소의 다음 원소 list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 원소의 다음 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 tail list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트 마지막 원소의 list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 가장 마지막 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev(struct list\_elem \*elem){} |
| **Parameter** | list\_elem 구조체 포인터 |
| **Return** | 인자로 받은 원소의 이전 원소 list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 원소의 이전 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 head list\_elem 구조체 포인타 |
| **Function** | 인자로 받은 리스트의 head를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 head list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 head를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 tail list\_elem 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert(struct list\_elem \*before struct list\_elem \*elem){} |
| **Parameter** | 원소를 삽입할 위치의 전 원소, 삽입할 원소의 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 첫 번째 인자 원소의 뒤에 두 번째 인자 원소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice(struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last){} |
| **Parameter** | 부분 리스트를 삽입할 위치의 전 원소, 삽입할 부분 리스트의 시작 원소, 삽입할 부분 리스트의 끝 원소 |
| **Return** | 없음 |
| **Function** | 첫 인자로 받은 리스트 원소에 두 번째 인자 원소부터 세 번째 인자 원소까지 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front(struct list \*list, struct list\_elem \*elem){} |
| **Parameter** | push\_front할 리스트 구조체 포인터, push\_front될 list\_elem 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 리스트의 맨 앞에 인자로 받은 원소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back(struct list \*list, struct list\_elem \*elem){} |
| **Parameter** | push\_back할 리스트 구조체 포인터, push\_back될 list\_elem 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 리스트의 맨 뒤에 인자로 받은 원소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove(struct list\_elem \*elem){} |
| **Parameter** | 삭제할 원소의 구조체 포인터 |
| **Return** | 삭제한 원소 다음 원소의 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 원소의 앞 원소와 뒤 원소를 연결하고, 뒤 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front(struct list \*list){} |
| **Parameter** | pop\_front할 리스트의 구조체 포인터 |
| **Return** | pop\_front된 원소의 구조체 포인터 |
| **Function** | 리스트의 맨 앞 원소를 pop한 뒤 해당 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back(struct list \*list){} |
| **Parameter** | pop\_back할 리스트의 구조체 포인터 |
| **Return** | pop\_back된 원소의 구조체 포인터 |
| **Function** | 리스트의 맨 뒤 원소를 pop한 뒤 해당 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 맨 앞 원소의 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 맨 앞 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 맨 뒤 원소의 구조체 포인터 |
| **Function** | 인자로 받은 리스트의 맨 뒤 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트의 크기 |
| **Function** | 리스트를 순회하며 리스트의 크기를 측정한 뒤 크기를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 리스트가 비어있으면 true, 아니면 false |
| **Function** | 인자로 받은 리스트가 비어있으면 true, 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse(struct list \*list){} |
| **Parameter** | 리스트 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 리스트의 순서를 뒤집는다. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted(struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux){} |
| **Parameter** | 정렬되었는지 확인할 구간의 시작 원소, 끝 원소, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 해당 구간이 강한 오름차순(<, increasing)으로 정렬되었으면 true, 아니면 false |
| **Function** | 해당 구간이 강한 오름차순(<, increasing)으로 정렬되었으면 true, 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem \*find\_end\_of\_run(struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux){} |
| **Parameter** | 약한 오름차순(<=, non-decreasing) 정렬을 확인할 구간의 시작 원소, 끝 원소, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 해당 구간에서 약한 오름차순(<=, non-decreasing) 정렬이 끝나는 마지막 원소의 구조체 포인터 |
| **Function** | 해당 구간을 a부터 탐색하여 약한 오름차순(<=, non-decreasing)이 끝나는 마지막 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void inplace\_merge(struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux){} |
| **Parameter** | 정렬삽입할 위치의 이전 원소 구조체 포인터 a0, 위치의 끝 원소이자 정렬삽입할 구간의 이전 원소 구조체 포인터 a1b0, 정렬삽입할 구간의 마지막 원소 구조체 포인터 b1, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 없음 |
| **Function** | a1b0부터 b1까지의 구간을 a0부터 a1b0까지의 구간까지 정렬삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort(struct list \*list, list\_less\_func \*less, void \*aux){} |
| **Parameter** | 정렬할 리스트 구조체 포인터, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 없음 |
| **Function** | 인자로 받은 리스트를 non-decreasing order로 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered(struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux){} |
| **Parameter** | 삽입 뒤 정렬할 리스트 구조체 포인터, 삽입될 리스트 원소 구조체 포인터, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 없음 |
| **Function** | 인자로 받은 리스트 원소를 리스트에 삽입한 뒤, 리스트를 non-decreasing order로 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique(struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux){} |
| **Parameter** | 리스트 구조체 포인터 list, 리스트 구조체 포인터 duplicates, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 없음 |
| **Function** | duplicates가 NULL이면 list의 중복 원소를 제거하고, 이외의 경우에는 duplicates에 중복 원소가 제거된 list를 덮어쓴다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max(struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 최대값을 계산한 리스트 구조체 포인터, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 최대값 원소의 구조체 포인터 |
| **Function** | 리스트를 순회하며 값이 최대인 원소를 찾은 뒤 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min(struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 최소값을 계산한 리스트 구조체 포인터, list\_less\_func 함수(additional implementation의 list\_less 함수), void형 포인터 aux |
| **Return** | 최소값 원소의 구조체 포인터 |
| **Function** | 리스트를 순회하며 값이 최소인 원소를 찾은 뒤 반환한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init(struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux) |
| **Parameter** | 해시테이블 구조체 포인터, 해시함수 hash\_hash\_func \*hash(additional implementation의 hash\_hash 함수, hash\_less\_func \*less(additional implementation의 hash\_less 함수, void형 포인터 \*aux) |
| **Return** | 새로 만든 해시테이블의 버켓이 NULL이면 false, 이외에는 true |
| **Function** | 새로운 해시테이블을 초기화한다. 해시테이블의 버킷이 문제 없이 생성됐으면, hash\_clear함수를 호출하여 초기화하고 true를 반환한다. 이외에는 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear(struct hash \*h, hash\_action\_func \*destructor){} |
| **Parameter** | 초기화할 해시테이블의 구조체 포인터, hash\_action\_func \*destructor(additional implementation의 hash\_destructor 함수) |
| **Return** | 없음 |
| **Function** | 인자로 받은 해시테이블을 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy |
| **Parameter** | 삭제할 해시테이블의 구조체 포인터, hash\_action\_func \*destructor(additional implementation의 hash\_destructor 함수) |
| **Return** | 없음 |
| **Function** | 인자로 받은 해시테이블을 삭제하고 메모리 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert(struct hash \*h, struct hash\_elem \*new){} |
| **Parameter** | 원소를 삽입할 해시테이블의 구조체 포인터, 삽입될 원소의 구조체 포인터 |
| **Return** | 삽입하려는 원소가 기존의 해시테이블에 존재하지 않았으면 NULL, 존재하면 해당 원소의 구조체 포인터 |
| **Function** | 삽입하려는 원소가 기존의 해시테이블에 존재하지 않는다면 해당 원소를 삽입하고 NULL을 반환, 존재한다면 삽입하지 않고 해당 원소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace(struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 원소를 삽입할 해시테이블의 구조체 포인터, 삽입될 원소의 구조체 포인터 |
| **Return** | 삽입하려는 원소가 기존의 해시테이블에 존재하지 않았으면 NULL, 존재하면 해당 원소의 구조체 포인터 |
| **Function** | 삽입하려는 원소가 기존의 해시테이블에 존재하지 않는다면 해당 원소를 삽입하고 NULL을 반환, 존재한다면 new로 교체한 뒤 기존의 원소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find(struct hash \*h, struct hash\_elem \*e){} |
| **Parameter** | 원소를 찾을 해시테이블의 구조체 포인터, 찾는 원소의 구조체 포인터 |
| **Return** | 찾는 원소가 해시테이블에 있으면 해당 원소 반환, 없으면 NULL 반환 |
| **Function** | 찾는 원소가 해시테이블에 있으면 해당 원소를 반환, 없으면 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete(struct hash \*h, struct hash\_elem \*e){} |
| **Parameter** | 원소를 삭제할 해시테이블의 구조체 포인터, 삭제할 원소의 구조체 포인터 |
| **Return** | 삭제하려는 원소가 존재한다면 해당 원소 반환, 존재하지 않는다면 NULL 반환 |
| **Function** | 삭제하려는 원소가 해시테이블에 존재하면 해시테이블에서 해당 원소를 삭제한 뒤 그 원소를 반환하고, 존재하지 않는다면 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | hash\_apply(struct hash \*h, hash\_action\_func \*action){} |
| **Parameter** | 데이터에 action의 작업을 할 해시테이블의 구조체 포인터, hash\_action\_func \*action(additional implementation에서 구현된 hash\_square 혹은 hash\_triple) |
| **Return** | 없음 |
| **Function** | 인자로 받은 해시테이블의 데이터에 action(square data or triple data)의 작업을 취한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first(struct hash\_iterator \*i, struct hash \*h){} |
| **Parameter** | hash\_iterator 구조체 포인터, 해시테이블 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 해시테이블의 반복자를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next(struct hash\_iterator \*i){} |
| **Parameter** | hash\_iterator 구조체 포인터 |
| **Return** | 해시테이블 반복자가 가리키는 원소 다음 원소의 구조체 포인터 |
| **Function** | 해시테이블 반복자를 다음 원소로 이동시키고, 다음 원소의 구조체 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur(struct hash\_iterator \*i) |
| **Parameter** | hash\_iterator 구조체 포인터 |
| **Return** | 해시테이블 반복자가 현재 가리키는 원소의 구조체 포인터 |
| **Function** | 해시테이블 반복자가 현재 가리키는 원소의 구조체 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size(struct hash \*h){} |
| **Parameter** | 크기를 계산할 해시테이블 구조체 |
| **Return** | 인자로 받은 해시테이블의 크기 |
| **Function** | 인자로 받은 해시테이블의 크기를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty(struct hash \*h){} |
| **Parameter** | 해시테이블의 구조체 포인터 |
| **Return** | 인자로 받은 해시테이블의 크기가 0이면 true, 0이 아니면 false |
| **Function** | 인자로 받은 해시테이블의 크기가 0이면 true, 0이 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes(const void \*buf\_, size\_t size){} |
| **Parameter** | 버퍼 buf, 크기 size |
| **Return** | 해시함수의 키 |
| **Function** | size byte에 따른 해시 키를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string(const char \*s\_) |
| **Parameter** | 문자열 s |
| **Return** | 해시함수의 키 |
| **Function** | 문자열에 해당하는 해시 키를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int(int i) |
| **Parameter** | 정수형 i |
| **Return** | 해시함수의 키 |
| **Function** | 인자로 받은 정수에 해당하는 해시 키를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \*find\_elem(struct hash \*h, struct list \*bucket, struct hash\_elem \*e){} |
| **Parameter** | 원소를 찾을 해시테이블의 구조체 포인터, 찾는 해시테이블 버킷의 구조체 포인터, 찾을 원소의 구조체 포인터 |
| **Return** | 찾는 원소가 존재하면 해당 원소, 존재하지 않는다면 NULL 반환 |
| **Function** | 찾는 원소가 존재하면 해당 원소, 존재하지 않는다면 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit(size\_t x) |
| **Parameter** | size\_t형 x |
| **Return** | x |
| **Function** | x의 가장 낮은 비트를 1에서 0로 설정한 뒤 x를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t is\_power\_of\_2(size\_t x) |
| **Parameter** | size\_t형 x |
| **Return** | x |
| **Function** | x가 2의 지수에 해당하면 true, 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void rehash(struct hash \*h){} |
| **Parameter** | 해시테이블 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 해시테이블의 버킷의 크기를 2의 지수의 크기로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem(struct hash \*h, struct list \*bucket, struct hash\_elem \*e){} |
| **Parameter** | 삽입할 해시테이블의 구조체 포인터, 삽입할 해시테이블의 버킷의 구조체 포인터, 삽입할 원소의 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 원소를 해시테이블의 버킷에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem(struct hash \*h, struct hash\_elem \*e){} |
| **Parameter** | 원소를 삭제할 해시테이블의 구조체 포인터, 삭제할 원소의 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 해시테이블의 버킷을 순회하며 인자로 받은 삭제할 원소를 발견해면 해당 원소를 삭제한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx(size\_t bit\_idx){} |
| **Parameter** | size\_t형 bit\_idx |
| **Return** | bit\_idx를 포함하는 원소의 인덱스 |
| **Function** | bit\_idx를 포함하는 원소의 인덱스를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask(size\_t bit\_idx){} |
| **Parameter** | size\_t형 bit\_idx |
| **Return** | bit\_idx의 값이 1인 원소 |
| **Function** | bit\_idx의 값이 1인 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt(size\_t bit\_cnt){} |
| **Parameter** | size\_t형 bit\_cnt |
| **Return** | bit\_cnt 크기를 표현할 때 필요한 bit의 수 |
| **Function** | bit\_cnt 크기를 표현할 때 필요한 bit의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t last\_mask(const struct bitmap \*b){} |
| **Parameter** | 비트맵 구조체 포인터 |
| **Return** | 가장 마지막 비트가 1이고 나머지가 0으로 비트마스킹된 elem\_type(unsigned long)형 정수 |
| **Function** | 인자로 받은 비트맵에서 가장 마지막 비트가 1이고 나머지가 0으로 비트마스킹된 elem\_type(unsigned long)형 정수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create(size\_t bit\_cnt){} |
| **Parameter** | 새로 만들 비트맵의 크기를 나타내는 정수 |
| **Return** | 새로 만들어진 비트맵 |
| **Function** | 인자로 들어온 bit\_cnt의 크기를 갖는 비트맵을 생성하고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf(size\_t bit\_cnt, void \*bloc, size\_t block\_size){} |
| **Parameter** | 새로 만들 비트맵의 크기를 나타내는 정수, void형 포인터 block, size\_t형 block\_size |
| **Return** | 새로 만들어진 비트맵 |
| **Function** | 인자로 들어온 bit\_cnt의 크기를 갖는 비트맵을 생성하고 반환한다. 해당 비트맵은 생성되기 전 block의 주소에 block\_size의 공간만큼 미리 할당된다 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size(size\_t bit\_cnt){} |
| **Parameter** | size\_t형 bit\_cnt |
| **Return** | 비트맵 구조체가 bit\_cnt의 크기를 가질 때 필요한 byte의 크기 |
| **Function** | 비트맵 구조체가 bit\_cnt의 크기를 가질 때 필요한 byte의 크기를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy(struct bitmap \*b){} |
| **Parameter** | 메모리 해제할 비트맵 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵을 메모리 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size(const struct bitmap \*b){} |
| **Parameter** | 크기를 계산할 비트맵 구조체 포인터 |
| **Return** | 인자로 받은 비트맵의 크기 |
| **Function** | 인자로 받은 비트맵의 크기를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set(struct bitmap \*b, size\_t idx, bool value){} |
| **Parameter** | 특정 비트의 값을 설정시킬 비트맵의 구조체 포인터, 설정되는 비트의 위치, 설정될 값 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵에서 인자로 받은 인덱스의 비트를 value값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark(struct bitmap \*b, size\_t bit\_idx){} |
| **Parameter** | 특정 비트를 1로 설정할 비트맵의 구조체 포인터, 1로 설정할 비트의 위치 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵의 bit\_idx 위치의 비트를 1로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset(struct bitmap \*b, size\_t bit\_idx){} |
| **Parameter** | 특정 비트를 0으로 설정할 비트맵의 구조체 포인터, 0으로 설정할 비트의 위치 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵의 bit\_idx 위치의 비트를 0으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip(struct bitmap \*b, size\_t bit\_idx){} |
| **Parameter** | 특정 비트를 반전시킬 비트맵의 구조체 포인터, 반전할 비트의 위치 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵의 bit\_idx 위치의 비트를 반전시킨다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test(const struct bitmap \*b, size\_t idx){} |
| **Parameter** | 특정 위치의 비트값을 확인할 비트맵 구조체, 비트값을 확인할 위치 |
| **Return** | 인자로 받은 비트맵에서 idx의 비트가 1이면 true, 0이면 false |
| **Function** | 인자로 받은 비트맵에서 idx의 비트가 1이면 true, 0이면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all(struct bitmap \*b, bool value){} |
| **Parameter** | 비트를 일괄설정할 비트맵 포인터, 일괄설정할 비트값 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵의 모든 값을 value값으로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple(struct bitmap \*b, size\_t start, size\_t cnt, bool value){} |
| **Parameter** | 특정 범위의 비트의 값을 수정할 비트맵 구조체 포인터, 값 수정 시작 위치, 수정할 비트의 수, 수정할 값 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵의 start번째 위치부터 cnt만큼 value의 값으로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count(const struct bitmap \*b, size\_t start, size\_t cnt, bool value){} |
| **Parameter** | 특정 값인 비트를 셀 비트맵의 구조체 포인터, 특정 값인 비트를 세는 위치, 특정 값인 비트를 셀 수, 찾는 값 |
| **Return** | 인자로 받은 비트맵에서 start부터 cnt만큼 특정 값을 가지는 비트의 수 |
| **Function** | 인자로 받은 비트맵의 start번째 위치부터 cnt만큼 탐색하면서 value의 값을 가지는 비트의 수를 세어 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains(const struct bitmap \*b, size\_t start, size\_t cnt, bool value){} |
| **Parameter** | 특정 값이 포함되는지 확인할 비트맵의 구조체 포인터, 확인을 시작할 비트의 위치, 확인하는 비트의 수, 확인할 값 |
| **Return** | 인자로 받은 비트맵의 start부터 cnt만큼 value의 값을 가지는 비트가 존재하면 true, 아니면 false |
| **Function** | 인자로 받은 비트맵의 start부터 cnt만큼 value의 값을 가지는 비트가 존재하면 true, 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any(const struct bitmap \*b, size\_t start, size\_t cnt){} |
| **Parameter** | 1인 비트가 있는지 탐색할 비트맵의 구조체 포인터, 탐색 시작 위치, 탐색 비트 수 |
| **Return** | 탐색하는 위치에 1인 비트가 있으면 true, 아니면 false |
| **Function** | 인자로 받은 비트맵의 start위치부터 cnt만큼의 구간에서 값이 1인 비트가 있으면 true, 없으면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none(const struct bitmap \*b, size\_t start, size\_t cnt){} |
| **Parameter** | 1인 비트가 있는지 탐색할 비트맵의 구조체 포인터, 탐색 시작 위치, 탐색 비트 수 |
| **Return** | 탐색하는 위치에 1인 비트가 없으면 true, 있으면 false |
| **Function** | 인자로 받은 비트맵의 start위치부터 cnt만큼의 구간에서 값이 1인 비트가 없으면 true, 있으면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all(const struct bitmap \*b, size\_t start, size\_t cnt){} |
| **Parameter** | 특정 구간의 모든 비트가 1인지 탐색할 비트맵의 구조체 포인터, 탐색 시작 위치, 탐색 비트 수 |
| **Return** | 탐색하는 위치의 모든 비트가 1이면 true, 0인 비트가 하나라도 있으면 false |
| **Function** | 인자로 받은 비트맵의 start위치부터 cnt만큼의 비트의 값이 모두 1이면 true, 0인 비트가 하나라도 있으면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan(const struct bitmap \*b, size\_t start, size\_t cnt, bool value){} |
| **Parameter** | value의 값을 연속적으로 갖는 구간이 있는지 탐색할 비트맵 구조체 포인터, 탐색 시작 위치, 탐색 비트 수, 탐색할 값 |
| **Return** | value의 값을 연속적으로 갖는 구간의 시작 인덱스, 해당 구간이 없으면 BITMAP\_ERROR |
| **Function** | 인자로 받은 비트맵의 start위치부터 cnt만큼 탐색하며 value의 값을 연속적으로 갖는 구간이 있는지 탐색한다. 해당 구간이 존재하면 구간의 시작 인덱스, 존재하지 않으면 BITMAP\_ERROR를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip(struct bitmap\*b, size\_t start, size\_t cnt, bool value){} |
| **Parameter** | value의 값을 연속적으로 갖는 구간이 있는지 탐색할 비트맵 구조체 포인터, 탐색 시작 위치, 탐색 비트 수, 탐색할 값 |
| **Return** | value의 값을 연속적으로 갖는 구간의 시작 인덱스, 해당 구간이 없으면 BITMAP\_ERROR |
| **Function** | 인자로 받은 비트맵의 start위치부터 cnt만큼 탐색하며 value의 값을 연속적으로 갖는 구간이 있는지 탐색한 뒤 탐색한 구간의 비트를 반전시킨다. 해당 구간이 존재하면 구간의 시작 인덱스, 존재하지 않으면 BITMAP\_ERROR를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size(const struct bitmap \*b){} |
| **Parameter** | 비트맵 구조체 포인터 |
| **Return** | 인자로 받은 비트맵을 저장하기 위해 필요한 byte수 |
| **Function** | 인자로 받은 비트맵을 저장하기 위해 필요한 byte수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read(struct bitmap \*b, struct file \*file){} |
| **Parameter** | 비트맵 구조체 포인터, 파일 구조체 포인터 |
| **Return** | 인자로 받은 파일에서 비트맵 b를 읽는 것에 성공하면 true, 실패하면 false |
| **Function** | 인자로 받은 파일에서 비트맵 b를 읽는다. 읽는 것에 성공하면 true, 실패하면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write(const struct bitmap \*b, struct file \*file){} |
| **Parameter** | 비트맵 구조체 포인터, 파일 구조체 포인터 |
| **Return** | 비트맵 b를 인자로 받은 파일에 쓰는 것에 성공하면 true, 실패하면 false |
| **Function** | 비트맵 b를 인자로 받은 파일에 쓴다. 쓰는 것에 성공하면 true, 실패하면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump(const struct bitmap \*b){} |
| **Parameter** | 출력할 비트맵 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 구조체를 16진수로 출력한다. |