**Project #1 : MyLib**

|  |  |
| --- | --- |
| Professor : | 박성용 교수님 |
| ID : | 20200152 |
| Name : | 김규빈 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | \*a, \*b: 위치를 바꿀 두 개의 노드 |
| **Return** | 없음 |
| **Function** | 두 개의 노드를 받아서 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | \*list: 순서를 무작위로 바꿀 리스트 |
| **Return** | 없음 |
| **Function** | 리스트 내의 원소 순서를 무작위로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | i: 변환할 int 값 |
| **Return** | unsigned type으로 변환된 결과 hash\_bytes를 통해 반환 |
| **Function** | 정수 i를 해싱하여, 정수에 기반한 고유한 해시 값을 생성 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | \*bitmap : expand할 bitmap,  size : expand할 크기 |
| **Return** | struct bitmap타입으로 expand된 bitmap을 반환 |
| **Function** | parameter로 받은 bitmap의 기존 사이즈에 size를 더해 expand |

|  |  |
| --- | --- |
| **Prototype** | int findStructureIndex(char c, char \*inst) |
| **Parameter** | c: 검색할 자료구조의 유형을 나타내는 문자, 'l'은 리스트, 'h'은 해시 테이블, 'b'는 비트맵  \*inst: 검색할 자료구조의 이름을 나타내는 문자열 포인터 |
| **Return** | 주어진 이름과 일치하는 자료구조의 인덱스를 int 타입으로 반환 |
| **Function** | 주어진 자료구조 유형(c)과 이름(inst)을 바탕으로 자료구조의 인덱스를 찾는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool compareListItems(const struct list\_elem \*a, const struct list\_elem \*b, void \*sub) |
| **Parameter** | \*a: 비교할 첫 번째 리스트 요소의 포인터.  \*b: 비교할 두 번째 리스트 요소의 포인터. |
| **Return** | 두 리스트 요소의 데이터 값을 비교하여, 첫 번째 데이터(a의 data)가 두 번째 데이터(b의 data)보다 작으면 true를, 그렇지 않으면 false를 반환 |
| **Function** | 첫 번째 데이터(a의 data)와 두 번째 데이터(b의 data) 비교 |

|  |  |
| --- | --- |
| **Prototype** | bool compareHashItems(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*sub) |
| **Parameter** | 비교하고자 하는 두개의 hash 요소 |
| **Return** | 만약 a 데이터가 b 데이터보다 작으면 true를 반환하고, 그렇지 않으면 false를 반환 |
| **Function** | a 데이터와 b 데이터 비교 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hashItemData(const struct hash\_elem \*e, void \*sub) |
| **Parameter** | \*e: 해싱할 해시 요소의 포인터 |
| **Return** | 해시 요소에 저장된 데이터를 해싱하여 unsigned int 타입의 해시 값을 반환 |
| **Function** | 해시 요소 e의 데이터를 특정 해시값으로 매핑하기 위해 hash\_int 함수를 호출 |

|  |  |
| --- | --- |
| **Prototype** | void square(struct hash\_elem \*e, void \*sub) |
| **Parameter** | \*e: 제곱할 데이터를 포함하고 있는 항목에 대한 hash\_elem 구조체를 가리키는 포인터 |
| **Return** | 없음 |
| **Function** | 해시 요소 e를 제곱 |

|  |  |
| --- | --- |
| **Prototype** | void triple(struct hash\_elem \*e, void \*sub) |
| **Parameter** | \*e: 세제곱할 데이터를 포함하고 있는 항목에 대한 hash\_elem 구조체를 가리키는 포인터 |
| **Return** | 없음 |
| **Function** | 해시 요소 e를 세제곱 |

|  |  |
| --- | --- |
| **Prototype** | void destructor(struct hash\_elem \*e, void \*sub) |
| **Parameter** | \*e: 소멸할 해시 요소의 포인터 |
| **Return** | 없음 |
| **Function** | 해시 요소 e 소멸 |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | #define list\_entry (list\_elem, struct, member) |
| **Parameter** | list\_elem: 리스트 엘리먼트의 포인터  struct: 상위 구조체의 타입  member: list\_elem이 포함된 상위 구조체 내의 멤버 이름 |
| **Return** | 상위 구조체의 포인터를 반환 |
| **Function** | 주어진 리스트 엘리먼트가 속한 상위 구조체의 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list\*) |
| **Parameter** | \*list: 초기화할 리스트의 포인터 |
| **Return** | 없음 |
| **Function** | 리스트 초기화, list를 인자로 받아 리스트의 시작과 끝을 나타내는 노드 설정 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin(struct list \*) |
| **Parameter** | \*list: 시작 요소를 검색할 리스트의 포인터 |
| **Return** | 리스트의 첫 번째 요소를 가리키는 struct list\_elem 포인터 |
| **Function** | 주어진 리스트의 첫 번째 요소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next(struct list\_elem \*elem) |
| **Parameter** | \*elem: 현재 list\_elem을 가리키는 포인터 |
| **Return** | 주어진 list\_elem 다음에 오는 원소에 대한 포인터를 반환 |
| **Function** | 리스트에서 주어진 원소 다음에 오는 원소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end(struct list \*) |
| **Parameter** | \*list: 끝 원소를 얻기 원하는 list 구조체의 포인터 |
| **Return** | 리스트의 끝을 나타내는 list\_elem 구조체에 대한 포인터를 반환 |
| **Function** | 리스트의 끝을 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev(struct list\_elem \*) |
| **Parameter** | \*elem: 현재 list\_elem을 가리키는 포인터 |
| **Return** | 주어진 list\_elem 이전에 오는 원소에 대한 포인터를 반환 |
| **Function** | 리스트에서 주어진 원소 이전에 오는 원소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list) |
| **Parameter** | \*list: ‘head’ 원소를 얻기 원하는 list 구조체의 포인터 |
| **Return** | 리스트의 ‘head’를 나타내는 list\_elem 구조체에 대한 포인터를 반환 |
| **Function** | 리스트의 'head' 원소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*list) |
| **Parameter** | \*list: ‘tail’ 원소를 얻기 원하는 list 구조체의 포인터 |
| **Return** | 리스트의 ‘tail’을 나타내는 list\_elem 구조체에 대한 포인터를 반환 |
| **Function** | 리스트의 'tail' 원소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | \*before: 새 원소가 삽입될 위치 바로 앞의 list\_elem 구조체를 가리키는 포인터  \*elem: 삽입될 새로운 list\_elem 구조체를 가리키는 포인터 |
| **Return** | 없음 |
| **Function** | 지정된 원소 before 앞에 새 원소 elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| **Parameter** | \*list: 원소를 삽입할 list 구조체의 포인터  \*elem: 삽입할 list\_elem 구조체의 포인터  \*less: 원소 비교를 위한 함수 포인터 |
| **Return** | 없음 |
| **Function** | 주어진 less 비교 함수를 사용하여 적절한 순서에 elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | \*before: 원소들이 삽입될 위치 바로 앞의 list\_elem 구조체를 가리키는 포인터  \*first: 삽입될 원소들의 범위 시작을 나타내는 list\_elem 구조체를 가리키는 포인터  \*last: 삽입될 원소들의 범위 끝 바로 다음을 나타내는 list\_elem 구조체를 가리키는 포인터 |
| **Return** | 없음 |
| **Function** | 리스트에서 first부터 last 바로 이전까지의 모든 원소들을 다른 위치인 before 앞으로 이동 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | \*list: 원소를 추가할 list 구조체의 포인터  \*elem: 리스트의 앞부분에 추가될 list\_elem 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | elem을 리스트의 맨 앞에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | \*list: 원소를 추가할 list 구조체의 포인터  \*elem: 리스트의 뒷부분에 추가될 list\_elem 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | elem을 리스트의 맨 뒤에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem) |
| **Parameter** | \*elem: 제거할 list\_elem 구조체의 포인터 |
| **Return** | 제거된 원소의 포인터를 반환 |
| **Function** | 주어진 elem을 리스트에서 제거 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list) |
| **Parameter** | \*list: 첫 번째 원소를 제거할 list 구조체의 포인터 |
| **Return** | 제거된 첫 번째 원소의 list\_elem 포인터를 반환 |
| **Function** | 리스트의 맨 앞 원소를 제거하고, 제거된 원소의 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list) |
| **Parameter** | \*list: 마지막 원소를 제거할 list 구조체의 포인터 |
| **Return** | 제거된 마지막 원소의 list\_elem 포인터를 반환 |
| **Function** | 리스트의 맨 뒤 원소를 제거하고, 제거된 원소의 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list) |
| **Parameter** | \*list: 첫 번째 원소의 포인터를 얻고자 하는 list 구조체의 포인터 |
| **Return** | 리스트의 첫 번째 원소의 list\_elem 포인터를 반환 |
| **Function** | 리스트의 첫 번째 원소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list) |
| **Parameter** | \*list: 마지막 원소의 포인터를 얻고자 하는 list 구조체의 포인터 |
| **Return** | 리스트의 마지막 원소의 list\_elem 포인터를 반환 |
| **Function** | 리스트의 마지막 원소를 가리키는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | \*list: 크기를 계산할 list 구조체의 포인터 |
| **Return** | 리스트에 있는 원소의 개수를 size\_t 타입으로 반환 |
| **Function** | 리스트에 포함된 원소의 총 개수를 계산하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*list) |
| **Parameter** | \*list: 비어 있는지 확인하고자 하는 list 구조체의 포인터 |
| **Return** | 리스트가 비어 있으면 true를, 그렇지 않으면 false를 반환 |
| **Function** | 리스트가 비어 있는지 여부를 확인 |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*list) |
| **Parameter** | \*list: 순서를 뒤집을 list 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | 리스트의 원소 순서를 뒤집음 |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | \*list: 정렬할 list 구조체의 포인터  \*less: 원소 비교를 위한 함수 포인터 |
| **Return** | 없음 |
| **Function** | less 함수를 사용하여 리스트를 정렬 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| **Parameter** | \*list: 중복을 제거할 list 구조체의 포인터  중복된 원소를 저장할 list 구조체의 포인터  \*less: 원소 비교를 위한 함수 포인터 |
| **Return** | 없음 |
| **Function** | 리스트에서 중복된 원소를 제거 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | \*list: 최소 원소를 찾을 list 구조체의 포인터  \*less: 원소 비교를 위한 함수 포인터 |
| **Return** | 가장 작은 원소의 list\_elem 포인터를 반환 |
| **Function** | less 함수를 사용하여 리스트에서 가장 작은 원소를 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | \*list: 최대 원소를 찾을 list 구조체의 포인터  \*less: 원소 비교를 위한 함수 포인터 |
| **Return** | 리스트에서 가장 큰 원소의 list\_elem 포인터를 반환 |
| **Function** | less 함수를 사용하여 리스트에서 가장 큰 원소를 찾아 반환 |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | void hash\_init (struct hash \*h, hash\_hash\_func \*hash,  hash\_less\_func \*less, void \*aux) |
| **Parameter** | \*h: 초기화할 해시 테이블의 포인터  \*hash: 해시 함수 포인터  \*less: 원소 비교 함수 포인터 |
| **Return** | 없음 |
| **Function** | 해시 테이블을 초기화 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | \*h: 작업을 수행할 해시 테이블의 포인터  \*action: 각 해시 테이블 원소에 적용할 함수의 포인터(square, triple) |
| **Return** | 없음 |
| **Function** | 해시 테이블 h의 모든 원소에 대해 action 함수를 적용 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | \*h: 해시 테이블을 비울 hash 구조체의 포인터  \*destructor: 각 해시 원소를 소멸시키기 위한 함수의 포인터 |
| **Return** | 없음 |
| **Function** | 해시 테이블 h의 모든 원소를 제거 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | \*h: 소멸시킬 해시 테이블의 hash 구조체 포인터  \*destructor: 각 해시 원소를 소멸시키기 위한 함수의 포인터 |
| **Return** | 없음 |
| **Function** | 해시 테이블 h 자체를 소멸시킴 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | \*h: 원소를 삽입할 해시 테이블의 hash 구조체 포인터  \*new: 해시 테이블에 삽입될 새로운 hash\_elem 구조체 포인터 |
| **Return** | 삽입 전에 해시 테이블에 이미 존재했던 동일한 키를 가진 원소의 hash\_elem 포인터를 반환, 만약 새 원소가 성공적으로 삽입되었다면 NULL을 반환 |
| **Function** | 새로운 원소 new를 해시 테이블 h에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | \*h: 원소를 교체할 해시 테이블의 hash 구조체 포인터  \*new: 해시 테이블에 교체될 새로운 hash\_elem 구조체 포인터 |
| **Return** | 해시 테이블 내에서 교체되기 전 원소의 hash\_elem 포인터를 반환, 만약 교체할 원소가 없었다면 NULL을 반환 |
| **Function** | 해시 테이블 h에 존재하는 원소를 새로운 원소 new로 교체 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | \*h: 검색을 수행할 해시 테이블의 hash 구조체 포인터  \*e: 찾고자 하는 원소의 hash\_elem 구조체 포인터 |
| **Return** | e와 동일한 키를 가진 해시 테이블 내의 원소의 hash\_elem 포인터를 반환 |
| **Function** | 해시 테이블 h에서 주어진 원소 e와 동일한 키를 가진 원소를 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | \*h: 원소를 제거할 해시 테이블의 hash 구조체 포인터  \*e: 제거할 원소의 hash\_elem 구조체 포인터 |
| **Return** | 제거된 원소의 hash\_elem 포인터를 반환 |
| **Function** | 해시 테이블 h에서 주어진 원소 e를 제거하고, 그 원소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | \*i: 해시 테이블을 순회할 hash\_iterator 구조체의 포인터  \*h: 순회를 시작할 해시 테이블의 hash 구조체 포인터 |
| **Return** | 없음 |
| **Function** | 해시 테이블 h의 순회를 위한 hash\_iterator i를 초기화 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i) |
| **Parameter** | \*i: 순회 중인 hash\_iterator 구조체의 포인터 |
| **Return** | 다음 hash\_elem 구조체에 대한 포인터를 반환 |
| **Function** | hash\_iterator i를 사용하여 해시 테이블의 현재 원소 다음에 오는 원소로 이동 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | \*i: 현재 원소를 가져올 hash\_iterator 구조체의 포인터 |
| **Return** | 현재 hash\_elem 구조체에 대한 포인터를 반환 |
| **Function** | hash\_iterator i에 의해 현재 가리켜지고 있는 해시 테이블 내의 원소의 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | \*h: 원소 개수를 확인할 hash 구조체의 포인터 |
| **Return** | 해시 테이블 h에 있는 원소의 개수를 size\_t 타입으로 반환 |
| **Function** | 해시 테이블 h에 현재 저장된 원소의 총 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h) |
| **Parameter** | \*h: 비어 있는지 확인할 hash 구조체의 포인터 |
| **Return** | 해시 테이블 h가 비어 있으면 true, 그렇지 않으면 false를 반환 |
| **Function** | 해시 테이블 h가 비어 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size) |
| **Parameter** | \*buf\_: 해싱할 데이터를 가리키는 포인터  size: buf\_에서 해싱할 데이터의 바이트 수 |
| **Return** | 해시된 값을 unsigned 타입으로 반환 |
| **Function** | 주어진 buf\_ 포인터에서 size 바이트만큼의 데이터를 해싱하여 그 결과를 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_) |
| **Parameter** | \*s\_: 해싱할 null-terminated 문자열의 포인터 |
| **Return** | 문자열 s\_의 해시 값을 unsigned 타입으로 반환 |
| **Function** | 문자열 s\_을 해싱하여, 문자열에 기반한 고유한 해시 값을 생성 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i) |
| **Parameter** | i: 해싱할 정수 |
| **Return** | 정수 i의 해시 값을 unsigned 타입으로 반환 |
| **Function** | 정수 i를 해싱하여, 정수에 기반한 고유한 해시 값을 생성 |

|  |  |
| --- | --- |
| **Prototype** | #define hash\_entry(hash\_elem, struct, member) |
| **Parameter** | hash\_elem: 해시 엘리먼트의 포인터  struct: 엘리먼트를 포함하는 구조체의 타입  member: 구조체 내 hash\_elem 멤버의 이름 |
| **Return** | 구조체의 포인터 |
| **Function** | hash\_elem으로부터 해당 엘리먼트를 포함하는 구조체의 포인터를 얻기 위해 사용 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | bit\_cnt: 생성할 비트맵의 비트 수 |
| **Return** | 새로 생성된 bitmap 구조체의 포인터를 반환 |
| **Function** | 지정된 수의 비트를 가진 비트맵을 생성 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | \*b: 소멸시킬 bitmap 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | bitmap\_destroy 함수로 생성된 비트맵 b를 소멸 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | \*b: 크기를 확인할 bitmap 구조체의 포인터 |
| **Return** | 비트맵 b의 비트 수를 size\_t 타입으로 반환 |
| **Function** | 비트맵 b에 포함된 비트의 총 수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | \*b: 비트 값을 설정할 bitmap 구조체의 포인터  idx: 설정할 비트의 인덱스  value: 설정할 값입니다. true면 비트를 1로 설정하고, false면 0으로 설정 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 idx 위치에 있는 비트를 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | \*b: 비트를 마킹할 bitmap 구조체의 포인터  bit\_idx: 마킹할 비트의 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 bit\_idx 위치에 있는 비트를 1로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | \*b: 비트를 리셋할 bitmap 구조체의 포인터  bit\_idx: 리셋할 비트의 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 bit\_idx 위치에 있는 비트를 0으로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | \*b: 비트를 반전시킬 bitmap 구조체의 포인터  bit\_idx: 반전시킬 비트의 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 bit\_idx 위치에 있는 비트의 값을 반전 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test(const struct bitmap \*b, size\_t idx) |
| **Parameter** | \*b: 테스트할 bitmap 구조체의 상수 포인터입니다.  idx: 테스트할 비트의 인덱스입니다. |
| **Return** | idx 위치에 있는 비트가 1이면 true, 0이면 false를 반환 |
| **Function** | 비트맵 b의 idx 위치에 있는 비트가 설정되었는지(1인지) 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | \*b: 모든 비트의 값을 설정할 bitmap 구조체의 포인터  value: 전체 비트에 설정할 값, true면 모든 비트를 1로 설정하고, false면 모든 비트를 0으로 설정 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 모든 비트를 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | \*b: 비트를 설정할 bitmap 구조체의 포인터  start: 값 설정을 시작할 비트의 인덱스  cnt: 설정할 비트의 개수  value: 설정할 값, true면 비트를 1로 설정하고, false면 0으로 설정 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 start 인덱스부터 cnt 개수만큼의 비트를 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | \*b: 비트를 계산할 bitmap 구조체의 상수 포인터  start: 계산을 시작할 비트의 인덱스  cnt: 계산할 비트의 개수  value: 계산에 포함할 비트의 값, true면 1인 비트를, false면 0인 비트를 계산 |
| **Return** | start 인덱스부터 cnt 개수만큼의 비트 중에서 value로 설정된 비트의 개수를 size\_t 타입으로 반환 |
| **Function** | 비트맵 b에서 start 인덱스부터 cnt 개수만큼의 비트 중에서 value 값과 일치하는 비트의 개수를 계산하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | \*b: 비트를 확인할 bitmap 구조체의 포인터  start: 확인을 시작할 비트의 인덱스  cnt: 확인할 비트의 개수  value: 확인할 비트의 값 |
| **Return** | start 인덱스부터 cnt 개수만큼의 비트 중에서 value로 설정된 비트가 하나라도 있으면 true, 전혀 없으면 false를 반환 |
| **Function** | 비트맵 b에서 start 인덱스부터 cnt 개수만큼의 비트 중에서 value 값으로 설정된 비트가 하나라도 있는지를 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | \*b: 검사할 bitmap 구조체의 상수 포인터  start: 검사를 시작할 첫 번째 비트의 인덱스  cnt: 검사할 비트의 개수 |
| **Return** | 지정된 범위 안에서 적어도 하나의 비트가 1로 설정되어 있으면 true, 모든 비트가 0이면 false를 반환 |
| **Function** | 비트맵 b에서 지정된 범위 start부터 cnt 개수만큼의 비트 중 적어도 하나가 1로 설정되어 있는지 여부를 확인 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | \*b: 덤프할 bitmap 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | 비트맵 b의 내용을 덤프 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | \*b: 검사할 bitmap 구조체의 상수 포인터  start: 검사를 시작할 첫 번째 비트의 인덱스  cnt: 검사할 비트의 개수 |
| **Return** | 지정된 범위 안의 모든 비트가 0이면 true, 하나라도 1이면 false를 반환 |
| **Function** | 비트맵 b의 지정된 범위 내의 모든 비트가 0으로 설정되어 있는지 여부를 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | \*b: 검사할 bitmap 구조체의 상수 포인터  start: 검사를 시작할 첫 번째 비트의 인덱스  cnt: 검사할 비트의 개수 |
| **Return** | 지정된 범위 안의 모든 비트가 1로 설정되어 있으면 true, 하나라도 0이면 false를 반환 |
| **Function** | 비트맵 b의 지정된 범위 내의 모든 비트가 1로 설정되어 있는지 여부를 확인 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | \*b: 스캔할 bitmap 구조체의 상수 포인터입니다.  start: 스캔을 시작할 첫 번째 비트의 인덱스입니다.  cnt: 스캔할 비트의 개수입니다.  value: 찾고자 하는 비트의 값 |
| **Return** | value를 가진 연속된 cnt 비트의 첫 번째 비트 인덱스를 size\_t 타입으로 반환 |
| **Function** | 비트맵 b에서 start 인덱스부터 시작하여 value를 가진 연속된 cnt 개수의 비트를 찾고, 찾은 경우 연속된 비트 그룹의 시작 인덱스를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | \*b: 스캔 및 반전할 bitmap 구조체의 포인터  start: 스캔을 시작할 첫 번째 비트의 인덱스  cnt: 스캔할 비트의 개수  value: 찾고자 하는 비트의 값 |
| **Return** | value를 가진 연속된 cnt 비트의 첫 번째 비트 인덱스를 size\_t 타입으로 반환 |
| **Function** | 비트맵 b에서 start 인덱스부터 시작하여 value로 지정된 값을 가진 연속된 cnt 개수의 비트를 찾고, 찾은 경우 이 비트들을 반전 |