**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 |
| 학번 : | 20221197 |
| 이름 : | 백서연 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | bool less(const struct list\_elem \*elem\_a, const struct list\_elem \*elem\_b, void \*aux); |
| **Parameter** | * **elem\_a**: 비교할 첫 번째 요소를 가리키는 포인터 * **elem\_b**: 비교할 두 번째 요소를 가리키는 포인터 * **aux**: 보조 매개 변수 (사용되지 않음) |
| **Return** | **bool** 타입. **elem\_a->data**가 **elem\_b->data**보다 작은 경우 **true**를, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | 두 요소의 데이터 필드를 비교하여 첫 번째 요소의 데이터가 두 번째 요소의 데이터보다 작은지를 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **unsigned hash\_my\_struct(const struct hash\_elem \*e, void \*aux)** |
| **Parameter** | * **e**: 해시 함수에 전달되는 해시 테이블 요소를 가리키는 포인터 * **aux**: 보조 매개 변수 (사용되지 않음) |
| **Return** | 반환값: **unsigned** 타입. **e**의 **data** 필드를 해싱하여 반환합니다. |
| **Function** | 기능: 해시 테이블의 요소를 해싱하여 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool hash\_less\_my\_struct(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux)** |
| **Parameter** | * **a**: 비교할 첫 번째 해시 테이블 요소를 가리키는 포인터 * **b**: 비교할 두 번째 해시 테이블 요소를 가리키는 포인터 * **aux**: 보조 매개 변수 (사용되지 않음) |
| **Return** | **bool** 타입. **a->data**가 **b->data**보다 작으면 **true**를, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | 두 해시 테이블 요소의 데이터 필드를 비교하여 첫 번째 요소의 데이터가 두 번째 요소의 데이터보다 작은지를 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void create\_hash(const char \*name)** |
| **Parameter** | **name**: 생성할 해시 테이블의 이름을 나타내는 문자열 |
| **Return** | 반환값: 없음 (**void**). 단순히 해시 테이블을 생성하고 초기화합니다. |
| **Function** | 기능: 주어진 이름으로 새로운 해시 테이블을 만들고 초기화합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void createBitmap(const char \*name, size\_t bit\_cnt)** |
| **Parameter** | * **name**: 생성할 비트맵의 이름을 나타내는 문자열 * **bit\_cnt**: 비트맵의 비트 수 |
| **Return** | 반환값: 없음 (**void**). 비트맵을 생성하고 초기화합니다. |
| **Function** | 기능: 주어진 이름과 비트 수로 새로운 비트맵을 만들고 초기화합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void create\_list(const char \*name)** |
| **Parameter** | **name**: 생성할 리스트의 이름을 나타내는 문자열 |
| **Return** | 반환값: 없음 (**void**). 리스트를 생성하고 초기화합니다. |
| **Function** | 기능: 주어진 이름으로 새로운 리스트를 만들고 초기화합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void initializeBitmapList()** |
| **Parameter** | 매개변수: 없음 |
| **Return** | 반환값: 없음 (**void**). 비트맵 배열을 초기화합니다. |
| **Function** | 기능: 전역 변수로 선언된 비트맵 배열을 모두 NULL로 초기화합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void initialize\_all\_lists()** |
| **Parameter** | 매개변수: 없음 |
| **Return** | 반환값: 없음 (**void**). 모든 리스트를 초기화합니다. |
| **Function** | 기능: 전역 변수로 선언된 리스트 배열을 모두 초기화합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void delete\_list(struct list \*my\_list)** |
| **Parameter** | **my\_list**: 삭제할 리스트를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**). 리스트의 모든 요소를 해제합니다 |
| **Function** | 기능: 주어진 리스트의 모든 요소를 해제하고 리스트를 비웁니다. |

|  |  |
| --- | --- |
| **Prototype** | **void dumpdata\_bitmap\_binary(const struct bitmap \*b)** |
| **Parameter** | **b**: 출력할 비트맵을 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**). 비트맵을 이진 형태로 출력합니다. |
| **Function** | 기능: 주어진 비트맵을 이진 형태로 출력합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void dumpdata\_hash(const struct hash \*h)** |
| **Parameter** | **h**: 출력할 해시 테이블을 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**). 해시 테이블의 데이터를 출력합니다. |
| **Function** | 기능: 주어진 해시 테이블의 데이터를 출력합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void square(struct hash\_elem \*e, void \*aux)** |
| **Parameter** | **e**: 제곱할 요소를 가리키는 해시 테이블 요소의 포인터  **aux**: 보조 매개 변수 (사용되지 않음) |
| **Return** | 반환값: 없음 (**void**). 요소의 데이터를 제곱합니다. |
| **Function** | 기능: 주어진 요소의 데이터를 제곱합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void triple(struct hash\_elem \*e, void \*aux)** |
| **Parameter** | * **e**: 세제곱할 요소를 가리키는 해시 테이블 요소의 포인터 * **aux**: 보조 매개 변수 (사용되지 않음) |
| **Return** | 반환값: 없음 (**void**). 요소의 데이터를 세제곱합니다. |
| **Function** | 기능: 주어진 요소의 데이터를 세제곱합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct hash\_elem \*find\_hash\_elem\_by\_value(struct hash \*h, int data\_value)** |
| **Parameter** | * **h**: 검색할 해시 테이블을 가리키는 포인터 * **data\_value**: 찾고자 하는 데이터 값 |
| **Return** | 반환값: **struct hash\_elem** 포인터. 데이터 값과 일치하는 요소를 찾으면 해당 요소의 포인터를 반환하고, 찾지 못한 경우 **NULL**을 반환합니다. |
| **Function** | 기능: 주어진 해시 테이블에서 데이터 값과 일치하는 요소를 찾아 해당 요소의 포인터를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void execute\_list\_insert\_command(struct list \*list, int insert\_position, int insert\_value)** |
| **Parameter** | * **list**: 삽입할 리스트를 가리키는 포인터 * **insert\_position**: 삽입할 위치의 인덱스 * **insert\_value**: 삽입할 값 |
| **Return** | 반환값: 없음 (**void**). 리스트의 지정된 위치에 값을 삽입합니다. |
| **Function** | 기능: 주어진 리스트의 지정된 위치에 주어진 값을 삽입합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_nth\_elem(struct list \*list, int n)** |
| **Parameter** | **list**: 리스트를 가리키는 포인터  **n**: 찾고자 하는 요소의 인덱스 |
| **Return** | 반환값: **struct list\_elem** 포인터. 리스트의 **n**번째 요소를 가리키는 포인터를 반환하고, 해당 인덱스가 리스트의 범위를 벗어난 경우 **NULL**을 반환합니다. |
| **Function** | 기능: 주어진 리스트의 지정된 인덱스에 해당하는 요소를 반환합니다. |

**필수로 구현하라고 했던 함수들**

|  |  |
| --- | --- |
| **Prototype** | **void list\_shuffle(struct list \*list)** |
| **Parameter** | 매개변수: **list** - 섞을 리스트를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트를 섞습니다. 리스트의 각 요소를 무작위로 섞어서 순서를 바꿉니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_swap(struct list\_elem \*a, struct list\_elem \*b)** |
| **Parameter** | **a**, **b**: 교환할 두 요소를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 두 리스트 요소를 교환합니다. 요소가 인접한 경우 직접 교환하고, 그렇지 않은 경우 요소들의 연결을 재구성하여 교환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **unsigned hash\_int\_2(int i)** |
| **Parameter** | **i** - 해싱할 정수 값 |
| **Return** | **unsigned** 타입. 해싱된 값 |
| **Function** | 주어진 정수를 해싱하여 반환합니다. 특정한 상수와 XOR 비트 연산을 사용하여 해시 값을 생성합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size)** |
| **Parameter** | * **bitmap**: 확장할 비트맵을 가리키는 포인터 * **size**: 확장할 크기 |
| **Return** | **struct bitmap** 포인터. 확장된 비트맵의 포인터를 반환하며, 실패할 경우  **NULL**을 반환합니다. |
| **Function** | 주어진 비트맵을 주어진 크기로 확장합니다. 새로운 크기가 현재 크기보다 작거나 같으면 확장하지 않습니다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | **void list\_init(struct list \*list)** |
| **Parameter** | **list** - 초기화할 리스트를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트를 초기화하고 빈 상태로 만듭니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_insert(struct list\_elem \*before, struct list\_elem \*elem)** |
| **Parameter** | * **before**: 삽입할 위치를 가리키는 포인터 * **elem**: 삽입할 요소를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 주어진 위치 **before** 이전에 새로운 요소 **elem**을 삽입합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_splice(struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last)** |
| **Parameter** | * **before**: 삽입할 위치를 가리키는 포인터 * **first**: 삽입할 요소들 중 첫 번째를 가리키는 포인터 * **last**: 삽입할 요소들 중 마지막 다음을 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트의 일부분을 다른 위치에 이동하거나 삽입합니다. |

|  |  |
| --- | --- |
| **Prototype** | 원형: **void list\_push\_front(struct list \*list, struct list\_elem \*elem)** |
| **Parameter** | * **list**: 리스트를 가리키는 포인터 * **elem**: 리스트의 앞쪽에 삽입할 요소를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트의 맨 앞에 요소를 추가합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_push\_back(struct list \*list, struct list\_elem \*elem)** |
| **Parameter** | * **list**: 리스트를 가리키는 포인터 * **elem**: 리스트의 뒤쪽에 삽입할 요소를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트의 맨 뒤에 요소를 추가합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_remove(struct list\_elem \*elem)** |
| **Parameter** | * **elem** - 제거할 요소를 가리키는 포인터 |
| **Return** | 반환값:제거된 요소의 다음 요소를 가리키는 포인터 |
| **Function** | 기능: 리스트에서 주어진 요소를 제거하고, 제거된 요소의 다음 요소를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_pop\_front(struct list \*list)** |
| **Parameter** | **list** - 요소를 제거할 리스트를 가리키는 포인터 |
| **Return** | 반환값: 제거된 맨 앞 요소를 가리키는 포인터 |
| **Function** | 기능: 리스트의 맨 앞 요소를 제거하고 해당 요소를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_pop\_back(struct list \*list)** |
| **Parameter** | **list** - 요소를 제거할 리스트를 가리키는 포인터 |
| **Return** | 반환값: 제거된 맨 뒤 요소를 가리키는 포인터 |
| **Function** | 기능: 리스트의 맨 뒤 요소를 제거하고 해당 요소를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_front(struct list \*list)** |
| **Parameter** | **list** - 요소를 조회할 리스트를 가리키는 포인터 |
| **Return** | 반환값: 리스트의 맨 앞 요소를 가리키는 포인터 |
| **Function** | 기능: 리스트의 맨 앞 요소를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_back(struct list \*list)** |
| **Parameter** | **list** - 요소를 조회할 리스트를 가리키는 포인터 |
| **Return** | 반환값: 리스트의 맨 뒤 요소를 가리키는 포인터 |
| **Function** | 기능: 리스트의 맨 뒤 요소를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_reverse(struct list \*list)** |
| **Parameter** | **list** - 역순으로 만들 리스트를 가리키는 포인터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트의 순서를 역순으로 뒤집습니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_sort(struct list \*list, list\_less\_func \*less, void \*aux)** |
| **Parameter** | * **list**: 정렬할 리스트를 가리키는 포인터 * **less**: 정렬 기준을 제공하는 함수 포인터 * **aux**: 정렬에 필요한 보조 데이터 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트를 주어진 정렬 기준에 따라 정렬합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_insert\_ordered(struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux)** |
| **Parameter** | * **list**: 정렬된 순서로 삽입할 리스트를 가리키는 포인터 * **elem**: 삽입할 요소를 가리키는 포인터 * **less**: 삽입 순서를 결정할 비교 함수 포인터 * **aux**: 비교 함수에 필요한 추가 매개변수 |
| **Return** | 없음 (**void**) |
| **Function** | 정렬된 순서를 유지하면서 요소를 삽입합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_unique(struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux)** |
| **Parameter** | * **list**: 중복 요소를 제거할 리스트를 가리키는 포인터 * **duplicates**: 중복된 요소를 저장할 리스트를 가리키는 포인터 * **less**: 중복 여부를 판단할 비교 함수 포인터 * **aux**: 비교 함수에 필요한 추가 매개변수 |
| **Return** | 반환값: 없음 (**void**) |
| **Function** | 기능: 리스트에서 중복된 요소를 제거하고, 필요한 경우 중복된 요소를 다른 리스트에 저장합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_max(struct list \*list, list\_less\_func \*less, void \*aux)** |
| **Parameter** | * **list**: 최댓값을 찾을 리스트를 가리키는 포인터 * **less**: 최댓값을 비교하는 함수 포인터 * **aux**: 비교 함수에 필요한 추가 매개변수 |
| **Return** | 반환값: 최댓값을 가리키는 요소 포인터 |
| **Function** | 기능: 리스트에서 최댓값을 가리키는 요소를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct list\_elem \*list\_min(struct list \*list, list\_less\_func \*less, void \*aux)** |
| **Parameter** | * **list**: 최솟값을 찾을 리스트를 가리키는 포인터 * **less**: 최솟값을 비교하는 함수 포인터 * **aux**: 비교 함수에 필요한 추가 매개변수 |
| **Return** | 반환값: 최솟값을 가리키는 요소 포인터 |
| **Function** | 기능: 리스트에서 최솟값을 가리키는 요소를 반환합니다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | **bool 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**: 비교 함수 포인터 * **aux**: 보조 매개변수 |
| **Return** | **반환값:** 초기화 성공 여부를 나타내는 **bool** 값 |
| **Function** | **기능:** 해시 테이블을 초기화하고 필요한 매개변수들을 설정합니다. 메모리 할당이 실패하면 **false**를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void hash\_clear(struct hash \*h, hash\_action\_func \*destructor)** |
| **Parameter** | * **h**: 초기화할 해시 테이블의 포인터 * **destructor**: 요소를 삭제할 때 호출할 함수 포인터 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 해시 테이블의 모든 요소를 제거하고, 필요한 경우 요소를 삭제할 때 사용될 함수를 호출합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void hash\_destroy(struct hash \*h, hash\_action\_func \*destructor)** |
| **Parameter** | * **h**: 제거할 해시 테이블의 포인터 * **destructor**: 요소를 삭제할 때 호출할 함수 포인터 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 해시 테이블을 제거하고 할당된 메모리를 해제합니다. 필요한 경우 요소를 삭제할 때 사용될 함수를 호출합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct hash\_elem \*hash\_insert(struct hash \*h, struct hash\_elem \*new)** |
| **Parameter** | * **h**: 삽입할 해시 테이블의 포인터 * **new**: 삽입할 새로운 해시 요소의 포인터 |
| **Return** | **반환값:** 삽입된 요소의 포인터 (기존에 이미 존재하던 경우 **NULL**) |
| **Function** | **기능:** 새로운 해시 요소를 테이블에 삽입합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct hash\_elem \*hash\_delete(struct hash \*h, struct hash\_elem \*e)** |
| **Parameter** | * **h**: 삭제할 해시 테이블의 포인터 * **e**: 삭제할 해시 요소의 포인터 |
| **Return** | **반환값:** 삭제된 요소의 포인터 (존재하지 않는 경우 **NULL**) |
| **Function** | **기능:** 주어진 해시 요소를 테이블에서 삭제합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void hash\_apply(struct hash \*h, hash\_action\_func \*action)** |
| **Parameter** | * **h**: 작업을 수행할 해시 테이블의 포인터 * **action**: 각 요소에 대해 수행할 작업을 나타내는 함수 포인터 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 각 요소에 대해 주어진 작업을 수행합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool hash\_empty(struct hash \*h)** |
| **Parameter** | **매개변수:** **h**: 확인할 해시 테이블의 포인터 |
| **Return** | **반환값:** 해시 테이블이 비어있는지 여부를 나타내는 **bool** 값 |
| **Function** | **기능:** 해시 테이블이 비어있는지 여부를 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **unsigned hash\_int(int i)** |
| **Parameter** | **매개변수:** **i**: 해싱할 정수 값 |
| **Return** | **반환값:** 해시된 정수 값 |
| **Function** | **기능:** 정수를 해싱하여 해시 값으로 변환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct hash\_elem \*hash\_find(struct hash \*h, struct hash\_elem \*e)** |
| **Parameter** | **h**: 해시 테이블의 포인터, **e**: 찾을 해시 요소의 포인터 |
| **Return** | **반환값:** 찾은 해시 요소의 포인터를 반환합니다. 요소를 찾지 못한 경우 **NULL**을 반환합니다. |
| **Function** | **기능:** 주어진 해시 테이블에서 주어진 해시 요소를 찾아서 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct hash\_elem \*hash\_delete(struct hash \*h, struct hash\_elem \*e)** |
| **Parameter** | **h**: 해시 테이블의 포인터, **e**: 삭제할 해시 요소의 포인터 |
| **Return** | **반환값:** 삭제된 해시 요소의 포인터를 반환합니다. 요소를 찾지 못한 경우 **NULL**을 반환합니다. |
| **Function** | **기능:** 주어진 해시 테이블에서 주어진 해시 요소를 찾아서 삭제하고, 그 결과를 반환합니다.. |

|  |  |
| --- | --- |
| **Prototype** | **size\_t hash\_size(struct hash \*h)** |
| **Parameter** | **h**: 해시 테이블의 포인터 |
| **Return** | **반환값:** 해시 테이블에 있는 요소의 개수를 반환합니다. |
| **Function** | **기능:** 주어진 해시 테이블에 있는 요소의 개수를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct hash\_elem \*hash\_replace(struct hash \*h, struct hash\_elem \*new)** |
| **Parameter** | **h**: 해시 테이블의 포인터, **new**: 삽입할 새로운 해시 요소의 포인터 |
| **Return** | 이전에 있던 같은 키를 가진 해시 요소의 포인터를 반환합니다. 새로운 요소를 삽입하는 경우 **NULL**을 반환합니다.. |
| **Function** | 주어진 해시 테이블에서 주어진 해시 요소를 찾아서 삭제한 후 새로운 해시 요소를 삽입하고, 이전 요소의 포인터를 반환합니다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_set(struct bitmap \*b, size\_t idx, bool value)** |
| **Parameter** | * **b**: 비트맵의 포인터 * **idx**: 설정할 비트의 인덱스 * **value**: 설정할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 주어진 인덱스에 해당하는 비트를 주어진 값으로 설정합니다. |

|  |  |
| --- | --- |
| **Prototype** | **struct bitmap \*bitmap\_create(size\_t bit\_cnt)** |
| **Parameter** | **bit\_cnt**: 생성할 비트맵의 비트 수 |
| **Return** | **반환값:** 생성된 비트맵의 포인터 |
| **Function** | **기능:** 주어진 비트 수로 비트맵을 생성합니다 |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_destroy(struct bitmap \*b)** |
| **Parameter** | **매개변수:** **b**: 제거할 비트맵의 포인터 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 주어진 비트맵을 제거하고 해당 메모리를 해제합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_mark(struct bitmap \*b, size\_t bit\_idx)** |
| **Parameter** | **b**: 비트맵의 포인터, **bit\_idx**: 설정할 비트의 인덱스 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 주어진 인덱스에 해당하는 비트를 설정합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_reset(struct bitmap \*b, size\_t bit\_idx)** |
| **Parameter** | **b**: 비트맵의 포인터, **bit\_idx**: 리셋할 비트의 인덱스 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 주어진 인덱스에 해당하는 비트를 리셋합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_flip(struct bitmap \*b, size\_t bit\_idx)** |
| **Parameter** | **b**: 비트맵의 포인터, **bit\_idx**: 플립할 비트의 인덱스 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 주어진 인덱스에 해당하는 비트를 플립(반전)합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_test(const struct bitmap \*b, size\_t idx)** |
| **Parameter** | **b**: 비트맵의 포인터, **idx**: 테스트할 비트의 인덱스 |
| **Return** | **반환값:** 해당 인덱스에 있는 비트의 값 (**true** 또는 **false**) |
| **Function** | **기능:** 주어진 인덱스에 해당하는 비트의 값을 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_set\_all(struct bitmap \*b, bool value)** |
| **Parameter** | **b**: 비트맵의 포인터, **value**: 설정할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 비트맵의 모든 비트를 주어진 값으로 설정합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_set\_multiple(struct bitmap \*b, size\_t start, size\_t cnt, bool value)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 설정을 시작할 인덱스, **cnt**: 설정할 비트의 개수, **value**: 설정할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 주어진 범위의 비트를 주어진 값으로 설정합니다. |

|  |  |
| --- | --- |
| **Prototype** | **size\_t bitmap\_count(const struct bitmap \*b, size\_t start, size\_t cnt, bool value)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 세어야 할 비트의 개수, **value**: 카운트할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 주어진 범위에서 주어진 값의 비트 개수 |
| **Function** | **기능:** 주어진 범위에서 주어진 값과 일치하는 비트의 개수를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_contains(const struct bitmap \*b, size\_t start, size\_t cnt, bool value)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 검사할 비트의 개수, **value**: 비교할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 주어진 범위에서 주어진 값의 비트가 하나라도 포함되어 있으면 **true**, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 주어진 값과 일치하는 비트가 하나라도 있는지 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_any(const struct bitmap \*b, size\_t start, size\_t cnt)** |
| **Parameter** | **매개변수:** **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 검사할 비트의 개수 |
| **Return** | **반환값:** 주어진 범위에서 하나 이상의 비트가 설정된 경우 **true**, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 하나 이상의 비트가 설정되어 있는지 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_none(const struct bitmap \*b, size\_t start, size\_t cnt)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 검사할 비트의 개수 |
| **Return** | **반환값:** 주어진 범위에서 모든 비트가 설정되지 않은 경우 **true**, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 모든 비트가 설정되어 있지 않은지 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_all(const struct bitmap \*b, size\_t start, size\_t cnt)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 검사할 비트의 개수 |
| **Return** | **반환값:** 주어진 범위에서 모든 비트가 설정된 경우 **true**, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 모든 비트가 설정되어 있는지 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **size\_t bitmap\_scan(const struct bitmap \*b, size\_t start, size\_t cnt, bool value)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 검색을 시작할 인덱스, **cnt**: 검색할 비트의 개수, **value**: 검색할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 주어진 범위에서 주어진 값과 일치하는 첫 번째 그룹의 시작 인덱스를 반환하고, 해당 그룹을 찾을 수 없는 경우 **BITMAP\_ERROR**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 주어진 값과 일치하는 첫 번째 그룹의 시작 인덱스를 찾습니다. |

|  |  |
| --- | --- |
| **Prototype** | **size\_t bitmap\_scan\_and\_flip(struct bitmap \*b, size\_t start, size\_t cnt, bool value)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 검색을 시작할 인덱스, **cnt**: 검색할 비트의 개수, **value**: 검색할 값 (**true** 또는 **false**) |
| **Return** | **반환값:** 주어진 범위에서 주어진 값과 일치하는 첫 번째 그룹의 시작 인덱스를 반환하고, 해당 그룹을 찾을 수 없는 경우 **BITMAP\_ERROR**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 주어진 값과 일치하는 첫 번째 그룹을 찾고, 해당 그룹의 비트를 반전시킵니다. |

|  |  |
| --- | --- |
| **Prototype** | **size\_t bitmap\_file\_size(const struct bitmap \*b)** |
| **Parameter** | **b**: 비트맵의 포인터 |
| **Return** | **반환값:** 비트맵을 파일에 저장하는 데 필요한 바이트 수를 반환합니다. |
| **Function** | **기능:** 비트맵을 파일에 저장하는 데 필요한 바이트 수를 반환합니다. |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_dump(const struct bitmap \*b)** |
| **Parameter** | **b**: 비트맵의 포인터 |
| **Return** | **반환값:** 없음 |
| **Function** | **기능:** 비트맵의 내용을 16진수로 콘솔에 덤프합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_all(const struct bitmap \*b, size\_t start, size\_t cnt)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 검사할 비트의 개수 |
| **Return** | **반환값:** 주어진 범위에서 모든 비트가 설정된 경우 **true**, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 모든 비트가 설정되어 있는지 확인하여 그 결과를 반환합니다. 이 함수는 **bitmap\_contains** 함수를 사용하여 모든 비트가 설정되지 않은 경우를 확인합니다. |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_all(const struct bitmap \*b, size\_t start, size\_t cnt)** |
| **Parameter** | **b**: 비트맵의 포인터, **start**: 시작 인덱스, **cnt**: 검사할 비트의 개수 |
| **Return** | **반환값:** 주어진 범위에서 모든 비트가 설정된 경우 **true**, 그렇지 않으면 **false**를 반환합니다. |
| **Function** | **기능:** 주어진 범위에서 모든 비트가 설정되어 있는지 확인하여 그 결과를 반환합니다. 이 함수는 **bitmap\_contains** 함수를 사용하여 모든 비트가 설정되어 있는지를 확인합니다. |