**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 |
| 학번 : | 20212020 |
| 이름 : | 박민준 |
|  |  |

**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* |
| **Return** | void |
| **Function** | 이중 연결 리스트에서 두 개의 요소를 교환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | void |
| **Function** | 주어진 리스트의 요소들을 무작위로 섞는다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int *i*); |
| **Parameter** | *i* |
| **Return** | unsigned |
| **Function** | hash\_int\_2 함수는 임의로 설정한 식에 의해 정수의 해시 값을 계산하여 반환하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \**b*, int *size*); |
| **Parameter** | *b, size* |
| **Return** | struct bitmap \* |
| **Function** | 비트맵의 크기를 확장하고, 그에 따라 메모리를 재할당한다. 또한 새로 추가된 비트를 초기화하여 false로 설정한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | Void list\_init (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | void |
| **Function** | 이중 연결 리스트를 초기화하고, 리스트의 head와 tail을 설정해 빈 리스트를 만든다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 시작 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \**elem*); |
| **Parameter** | *elem* |
| **Return** | struct list\_elem \* |
| **Function** | 주어진 요소의 다음 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 끝을 나타내는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 역순에서 시작하는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \**elem*); |
| **Parameter** | *elem* |
| **Return** | struct list\_elem \* |
| **Function** | 주어진 요소의 이전 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 역순에서 끝을 나타내는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 시작을 가리키는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 끝을 가리키는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \**before*, struct list\_elem \**elem*); |
| **Parameter** | *before, elem* |
| **Return** | void |
| **Function** | 이중 연결 리스트의 before 요소 이전 위치에 elem 요소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \**before*, struct list\_elem \**first*, struct list\_elem \**last*); |
| **Parameter** | *before, first, last* |
| **Return** | void |
| **Function** | [first, last] 범위의 요소를 before 요소의 이전 위치로 이동시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \**list*, struct list\_elem \**elem*); |
| **Parameter** | *list, elem* |
| **Return** | void |
| **Function** | 이중 연결 리스트의 맨 앞에 요소를 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \**list*, struct list\_elem \**elem*); |
| **Parameter** | *list, elem* |
| **Return** | void |
| **Function** | 이중 연결 리스트의 맨 뒤에 요소를 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \**elem*); |
| **Parameter** | *elem* |
| **Return** | struct list\_elem \* |
| **Function** | 주어진 요소를 이중 연결 리스트에서 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 맨 앞 요소를 제거하고 해당 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 맨 뒤 요소를 제거하고 해당 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 맨 앞에 있는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | struct list\_elem \* |
| **Function** | 이중 연결 리스트의 맨 뒤에 있는 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | size\_t |
| **Function** | 이중 연결 리스트에 있는 요소의 전체 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | bool |
| **Function** | 이중 연결 리스트가 비어있는지 여부를 boolean 형태로 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\**a*, struct list\_elem \*\**b*); |
| **Parameter** | *a, b* |
| **Return** | void |
| **Function** | 두 개의 ‘struct list\_elem \*’을 가리키는 포인터를 교환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \**list*); |
| **Parameter** | *list* |
| **Return** | void |
| **Function** | 이중 연결 리스트의 순서를 역순으로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \**a*, struct list\_elem \**b*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *a, b, less, aux* |
| **Return** | bool |
| **Function** | 주어진 범위의 리스트 요소가 주어진 비교 함수(list\_less\_func)에 따라 정렬되어 있는지 여부를 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** | *a, b, less, aux* |
| **Return** | static struct list\_elem \* |
| **Function** | 정렬된 리스트의 주어진 범위에서 연속된 요소들의 끝을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | static void inplace\_merge (struct list\_elem \**a0*, struct list\_elem \**a1b0*, struct list\_elem \**b1*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *a0, a1b0, b1, less, aux* |
| **Return** | void |
| **Function** | 두 개의 정렬된 리스트를 하나로 병합한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \**list*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *list, less, aux* |
| **Return** | void |
| **Function** | 주어진 리스트를 주어진 비교 함수(list\_less\_func)에 따라 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \**list*, struct list\_elem \**elem*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *list, elem, less, aux* |
| **Return** | void |
| **Function** | 정렬되어 있는 리스트에 주어진 요소를 삽입한다. 삽입 후에도 정렬을 유지한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \**list*, struct list \**duplicates*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *list, duplicates, less, aux* |
| **Return** | void |
| **Function** | 주어진 리스트에서 연속된 중복 요소를 제거한다. 중복된 요소들은 전부 duplicates 리스트에 삽입된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \**list*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *list, less, aux* |
| **Return** | struct list\_elem \* |
| **Function** | 리스트에서 가장 큰 값을 가진 요소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \**list*, list\_less\_func \**less*, void \**aux*); |
| **Parameter** | *list, less, aux* |
| **Return** | struct list\_elem \* |
| **Function** | 리스트에서 가장 작은 값을 가진 요소를 반환한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \**h*, hash\_hash\_func \**hash*, hash\_less\_func \**less*, void \**aux*); |
| **Parameter** | *h, hash, less, aux* |
| **Return** | bool |
| **Function** | 해시 테이블을 초기화하고, 해시 값을 계산하기 위해 hash 함수를 사용한다. 또한, 비교 함수 포인터 less와 보조 데이터 aux를 사용해 해시 요소를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \**h*, hash\_action\_func \**destructor*); |
| **Parameter** | *h, destructor* |
| **Return** | void |
| **Function** | 해시 테이블의 모든 요소를 제거한다. 요소를 삭제할 때 destructor가 지정되면 해당 함수를 호출하여 요소를 제거한다. 여기서 destructor는 요소를 제거할 때 호출할 함수 포인터이다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \**h*, hash\_action\_func \**destructor*); |
| **Parameter** | *h, destructor* |
| **Return** | void |
| **Function** | 해시 테이블을 파괴하고, destructor라는 action 함수가 지정되면 해당 함수를 호출하여 모든 요소를 파괴한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \**h*, struct hash\_elem \**new*); |
| **Parameter** | *h, new* |
| **Return** | struct hash\_elem \* |
| **Function** | 해시 테이블에 새로운 요소를 삽입한다. 중복된 요소가 있는 경우 삽입하지 않고 해당 요소를 반환하고, 중복된 요소가 없으면 NULL 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \**h*, struct hash\_elem \**new*); |
| **Parameter** | *h, new* |
| **Return** | struct hash\_elem \* |
| **Function** | 해시 테이블에 새로운 요소를 삽입하거나, 이미 있는 요소를 대체한다. 대체되는 요소가 있으면 해당 요소를 반환하고, 그렇지 않으면 NULL 포인터를 반환한다. 인자 h는 삽입 또는 대체할 해시 테이블이고, 인자 new는 삽입 또는 대체할 해시의 요소이다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \**h*, struct hash\_elem \**e*); |
| **Parameter** | *h, e* |
| **Return** | struct hash\_elem \* |
| **Function** | 해시 테이블에서 주어진 요소와 동일한 요소를 찾아 반환한다. 찾은 요소가 있으면 해당 요소를 반환하고, 그렇지 않으면 NULL 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \**h*, struct hash\_elem \**e*); |
| **Parameter** | *h, e* |
| **Return** | struct hash\_elem \* |
| **Function** | 해시 테이블에서 주어진 요소를 찾아 제거하고 반환한다. 삭제된 요소가 있으면 해당 요소를 반환하고, 그렇지 않으면 NULL 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \**h*, hash\_action\_func \**action*); |
| **Parameter** | *h, action* |
| **Return** | void |
| **Function** | 해시 테이블에서 각 요소에 대해 지정된 작업을 수행한다. h는 적용할 해시 테이블이고, action은 각 요소에 대해 호출할 함수 포인터이다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \**i*, struct hash \**h*); |
| **Parameter** | *i, h* |
| **Return** | void |
| **Function** | 해시 테이블을 반복하기 위한 반복자를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \**i*); |
| **Parameter** | *i* |
| **Return** | struct hash\_elem \* |
| **Function** | 반복자를 사용하여 해시 테이블에서 다음 요소로 이동한다. 해당 함수는 다음 요소를 반환하거나, 더 이상 요소가 없으면 NULL 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \**i*); |
| **Parameter** | *i* |
| **Return** | struct hash\_elem \* |
| **Function** | 반복 중인 해시 테이블에서 현재 요소를 반환하거나, 반복이 끝난 경우 NULL 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \**h*); |
| **Parameter** | *h* |
| **Return** | size\_t |
| **Function** | 해시 테이블에 있는 요소의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \**h*); |
| **Parameter** | *h* |
| **Return** | bool |
| **Function** | 해시 테이블이 비어 있는지 여부를 확인한다. 비어 있으면 true를 반환하고, 그렇지 않으면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \**buf\_*, size\_t *size*); |
| **Parameter** | *buf\_, size* |
| **Return** | unsigned |
| **Function** | 버퍼의 byte에 대한 해시 값을 계산하여 반환한다. buf\_는 해시를 계산할 버퍼이며, size는 해시를 계산할 버퍼의 크기이다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \**s\_*); |
| **Parameter** | *s\_* |
| **Return** | unsigned |
| **Function** | 문자열의 해시 값을 계산하여 반환한다. s\_는 해시를 계산할 문자열이다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int *i*); |
| **Parameter** | *i* |
| **Return** | unsigned |
| **Function** | 정수의 해시 값을 계산하여 반환한다. i는 해시를 계산할 정수이다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list \*find\_bucket (struct hash \**h*, struct hash\_elem \**e*); |
| **Parameter** | *h, e* |
| **Return** | static struct list \* |
| **Function** | 주어진 해시 테이블 h에서 요소 e가 속한 bucket을 찾아 반환한다. 해시 함수를 사용하여 요소의 해시 값을 계산하고, bucket의 인덱스를 계산한 후에 해당 버킷의 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \*find\_elem (struct hash \**h*, struct list \**bucket*, struct hash\_elem \**e*); |
| **Parameter** | *h, bucket, e* |
| **Return** | static struct hash\_elem \* |
| **Function** | 주어진 해시 테이블 h에서 bucket의 요소 e와 동일한 해시 요소를 찾는다. bucket에서 순차적으로 각 요소를 검사하여 찾고자 하는 요소와 동일한지 확인한 후, 동일한 요소를 찾으면 해당 요소를 반환하고 찾지 못하면 NULL 포인터를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \**h*); |
| **Parameter** | *h* |
| **Return** | void |
| **Function** | 해시 테이블의 bucket 수를 이상적인 값으로 조정한다. 해시 테이블에 저장된 요소 수를 고려하여 새로운 bucket 수를 계산하고, 필요한 경우 bucket 수를 조정한다. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \**h*, struct list \**bucket*, struct hash\_elem \**e*); |
| **Parameter** | *h, bucket, e* |
| **Return** | void |
| **Function** | 주어진 해시 테이블 h의 특정 bucket에 요소 e를 삽입한다. 요소를 삽입한 후에 해시 테이블의 요소 수를 증가시킨다. |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \**h*, struct hash\_elem \**e*); |
| **Parameter** | *h, e* |
| **Return** | void |
| **Function** | 해시 테이블 h에서 요소 e를 제거한다. 요소를 제거한 후에 해시 테이블의 요소 수를 감소시킨다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t *bit\_cnt*); |
| **Parameter** | *bit\_cnt* |
| **Return** | struct bitmap \* |
| **Function** | 지정된 비트 수로 새 비트맵을 동적으로 할당하고 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t *bit\_cnt*, void \**block*, size\_t *block\_size*); |
| **Parameter** | *bit\_cnt, block, block\_size* |
| **Return** | struct bitmap \* |
| **Function** | 미리 할당된 메모리 블록 내에서 비트맵을 생성하고 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \**b*); |
| **Parameter** | *b* |
| **Return** | void |
| **Function** | 비트맵과 할당된 메모리를 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \**b*); |
| **Parameter** | *b* |
| **Return** | size\_t |
| **Function** | 비트맵의 비트 수를 반환하다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \**b*, size\_t *idx*, bool *value*); |
| **Parameter** | *b, idx, value* |
| **Return** | void |
| **Function** | 비트맵에서 지정된 인덱스의 비트를 지정된 값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \**b*, size\_t *bit\_idx*); |
| **Parameter** | *b, bit\_idx* |
| **Return** | void |
| **Function** | 비트맵에서 지정된 인덱스의 비트를 true로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \**b*, size\_t *bit\_idx*); |
| **Parameter** | *b, bit\_idx* |
| **Return** | void |
| **Function** | 비트맵에서 지정된 인덱스의 비트를 false로 재설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \**b*, size\_t *bit\_idx*); |
| **Parameter** | *b, bit\_idx* |
| **Return** | void |
| **Function** | 비트맵에서 지정된 인덱스의 true면 false로, fasle면 true로 비트를 뒤집는다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \**b*, size\_t *idx*); |
| **Parameter** | *b, idx* |
| **Return** | bool |
| **Function** | 비트맵에서 지정된 인덱스의 비트 값을 반환한다. b는 비트를 테스트할 비트맵에 대한 포인터이고, idx는 테스트할 비트의 인덱스이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \**b*, bool *value*); |
| **Parameter** | *b, value* |
| **Return** | void |
| **Function** | 비트맵의 모든 비트를 지정된 값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \**b*, size\_t *start*, size\_t *cnt*, bool *value*); |
| **Parameter** | *b, start, cnt, value* |
| **Return** | void |
| **Function** | 비트맵에서 시작 인덱스부터 지정된 수의 비트를 지정된 값으로 설정한다. cnt는 설정할 비트의 수를 의미한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \**b*, size\_t *start*, size\_t *cnt*, bool *value*); |
| **Parameter** | *b*, cnt, value |
| **Return** | size\_t |
| **Function** | 비트맵에서 지정된 범위 내에서 지정된 값으로 설정된 비트의 수를 반환한다. cnt는 count할 비트의 수를 의미한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \**b*, size\_t *start*, size\_t *cnt*, bool *value*); |
| **Parameter** | *b, start, cnt, value* |
| **Return** | bool |
| **Function** | 비트맵에서 지정된 범위 내에서 지정된 값으로 설정된 비트가 있는지를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \**b*, size\_t *start*, size\_t *cnt*); |
| **Parameter** | *b, start, cnt* |
| **Return** | bool |
| **Function** | 비트맵에서 지정된 범위 내에서 true로 설정된 비트가 단 한 개라도 있는지를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \**b*, size\_t *start*, size\_t *cnt*); |
| **Parameter** | *b, start, cnt* |
| **Return** | bool |
| **Function** | 비트맵에서 지정된 범위 내에서 false로 설정된 비트가 단 한 개라도 있는지를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \**b*, size\_t *start*, size\_t *cnt*); |
| **Parameter** | *b, start, cnt* |
| **Return** | bool |
| **Function** | 비트맵에서 지정된 범위 내의 모든 비트가 true로 설정되어 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \**b*, size\_t *start*, size\_t *cnt*, bool *value*); |
| **Parameter** | *b, start, cnt, value* |
| **Return** | size\_t |
| **Function** | 지정된 값으로 설정된 첫 번째 그룹의 연속적인 비트를 검색하고 해당 그룹의 첫 번째 인덱스를 반환한다. 만약 그룹이 없는 경우 BITMAP\_ERROR 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \**b*, size\_t *start*, size\_t *cnt*, bool *value*); |
| **Parameter** | *b, start, cnt, value* |
| **Return** | size\_t |
| **Function** | 지정된 값으로 설정된 첫 번째 그룹의 연속적인 비트를 검색하고 해당 그룹의 모든 비트를 뒤집는다. 만약 그룹이 없는 경우 BITMAP\_ERROR 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \**b*); |
| **Parameter** | *b* |
| **Return** | size\_t |
| **Function** | 파일에 비트맵을 저장하는 데 필요한 바이트 수를 반환한다. 비트맵의 비트 수에 따라 바이트 수가 결정된다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \**b*); |
| **Parameter** | *b* |
| **Return** | void |
| **Function** | 비트맵의 내용을 콘솔에 16진수로 덤프한다. 비트맵의 각 비트에 대해 해당하는 16진수 값을 표시한다. |