**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 |
| 학번 : | 20xxxxxx |
| 이름 : |  |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | struct list\_elem \*a, struct list\_elem \*b |
| **Return** | None (void) |
| **Function** | 두 list\_elem의 doubly linked list 상에서 next, prev 포인터를 이이용해 위치를 서로 바꿔주는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | None (void) |
| **Function** | Doubly linked list를 랜덤하게 섞는 함수. List\_elem의 주소를 담는 배열을 선언하여 knuth shuffle algorithm 수행 후 순서에 맞게 다시 요소들을 연결하는 방식 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | int i |
| **Return** | Unsigned (hash값) |
| **Function** | Knuth multiplicative hash를 사용하여 해쉬값을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | struct bitmap \*bitmap, int size |
| **Return** | struct bitmap \* (확장된 bitmap의 포인터) |
| **Function** | 새로운 size를 인자로 받아 byte\_cnt를 통해 할당 할 크기 계산하고 malloc하여 bitmap의 bits을 복사. 기존의 bits는 메모리 해제 후 새로운 bits를 가리키게 한 후 bitmap return. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux |
| **Return** | Bool (true or false) |
| **Function** | Less 함수에 따라 두 list\_elem이 정렬되었는지 여부를 return |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head (struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | Bool (true or false) |
| **Function** | List\_elem이 list의 head인지 여부를 return |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior (struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | Bool (true or false) |
| **Function** | List\_elem의 prev와 next를 확인하여 list에 속하였는지 여부를 return |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail (struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | Bool (true or false) |
| **Function** | List\_elem이 list의 tail인지 여부를 return |

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | None (void) |
| **Function** | List 포인터를 인자로 받아 empty list로 초기화 시켜주는 함수 |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* (list의 첫번째 element) |
| **Function** | List 포인터를 인자로 받아 첫 번째 element (head의 next)를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | struct list\_elem \* (인자의 다음 element) |
| **Function** | List\_elem 포인터를 인자로 받아 다음 위치의 list\_elem 포인터를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* (list의 tail) |
| **Function** | List 포인터를 인자로 받아 tail을 return하는 함수 (마지막 요소의 next) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* (마지막 element) |
| **Function** | List의 포인터를 인자로 받아 마지막 element를 return하는 함수 (tail의 prev) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | struct list\_elem \* |
| **Function** | List\_elem 포인터를 인자로 받아 이전 위치의 element를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List 포인터를 인자로 받아 첫 번째 element를 return하는 함수 (head->next). List\_rbegin과 함께 reverse iteration에 사용됨 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List 포인터를 인자로 받아 head를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List 포인터를 인자로 받아 tail을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*before, struct list\_elem \*elem |
| **Return** | None (void) |
| **Function** | 두 list\_elem 포인터를 인자로 받아 list 상에서 before 바로 전 위치에 elem을 삽입하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice(struct list\_elem \*, struct list\_elem \*, struct list\_elem \*) |
| **Parameter** | struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last |
| **Return** | None (void) |
| **Function** | first부터 last까지의 elem을 before 바로 전 위치에 삽입하는 함수. First, last는 같은 list에 속하고 before는 다른 list일 수 있음. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | struct list \*list, struct list\_elem \*elem |
| **Return** | None (void) |
| **Function** | Elem을 list의 첫 번째 위치에 insert하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | struct list \*list, struct list\_elem \*elem |
| **Return** | None (void) |
| **Function** | Elem을 list의 마지막 위치에 insert하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | struct list\_elem \* |
| **Function** | Elem을 list에서 제거하면서 해당 elem의 포인터를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List의 첫 번째 elem을 제거하면서 해당 elem의 포인터를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List의 마지막 elem을 제거하면서 해당 elem의 포인터를 return 하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List의 첫 번째 elem에 대한 포인터를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | struct list\_elem \* |
| **Function** | List의 마지막 elem에 대한 포인터를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | size\_t |
| **Function** | List의 head, tail을 제외한 elem 개수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | bool |
| **Function** | List에 head, tail을 제외한 elem이 존재하는지 여부를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b) |
| **Parameter** | struct list\_elem \*\*a, struct list\_elem \*\*b |
| **Return** | void |
| **Function** | 두 list\_elem의 포인터의 reference를 받아와서 가리키는 위치를 서로 교환하여 swap하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*list) |
| **Parameter** | struct list \*list |
| **Return** | void |
| **Function** | List의 elem의 순서를 반대로 만드는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem \*find\_end\_of\_run (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux |
| **Return** | struct list\_elem \* |
| **Function** | List\_elem a 부터 시작하여 b 이전의 nondecreasing order인 요소들을 찾음. Run이 끝난 마지막 요소를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static void inplace\_merge (struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux |
| **Return** | void |
| **Function** | A0~A1B0과 A1B0~B1을 병합하는 함수. LESS 함수에 따라 nondecreasing order로 정렬됨 |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list \*list, list\_less\_func \*less, void \*aux |
| **Return** | void |
| **Function** | List를 LESS 함수 그리고 보조 데이터(aux)에 따라 sorting하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux |
| **Return** | void |
| **Function** | 인자로 들어오는 list는 반드시 정렬되있어야함. List\_elem을 LESS 함수 그리고 보조 데이터(aux)에 따라 정렬이 유지되는 적절한 위치에 삽입하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux |
| **Return** | void |
| **Function** | List의 elements에서 중복된 요소들을 제거하는 함수. Duplicates가 null이 아니라면 중복된 요소들은 duplicates에 추가됨 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list \*list, list\_less\_func \*less, void \*aux |
| **Return** | struct list\_elem \* |
| **Function** | list에서 가장 작은 값을 가진 elem을 return하는 함수. 여러 개라면 가장 먼저 나타나는 elem. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | struct list \*list, list\_less\_func \*less, void \*aux |
| **Return** | struct list\_elem \* |
| **Function** | list에서 가장 큰 값을 가진 elem을 return하는 함수. 여러 개라면 가장 먼저 나타나는 elem. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux) |
| **Parameter** | struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux |
| **Return** | bool |
| **Function** | 인자로 들어온 hash를 초기화하는 함수. Buckets에 대한 동적 할당이 실패하면 false return, 성공하면 true를 return. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | struct hash \*h, hash\_action\_func \*destructor |
| **Return** | void |
| **Function** | Hash의 모든 bucket을 순회하며 elem을 제거하는 함수. Hash\_elem 제거에 대한 behavior는 destructor 함수에 따라 정의됨 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | struct hash \*h, hash\_action\_func \*destructor |
| **Return** | void |
| **Function** | Hash\_clear를 수행하고 buckets에 대한 동적 할당을 해제하여 hash를 파괴시키는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | struct hash \*h, struct hash\_elem \*new |
| **Return** | struct hash\_elem \* |
| **Function** | Hash\_elem new를 hash h에 삽입하는 함수. 만약 중복되는 요소가 존재하면 기존에 있던 hash\_element를 return하고, 없다면 null을 return함 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | struct hash \*h, struct hash\_elem \*new |
| **Return** | struct hash\_elem \* |
| **Function** | Hash\_elem new를 hash h에 삽입하는 함수. 중복되는 요소가 있다면 new로 대체하고 기존 요소를 return함 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | struct hash \*h, struct hash\_elem \*e |
| **Return** | struct hash\_elem \* |
| **Function** | Hash\_elem e와 동일한 요소가 hash h에 존재하는지 검색하고 있다면 그 요소를 return하고 없다면 null을 return함 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | struct hash \*h, struct hash\_elem \*e |
| **Return** | struct hash\_elem \* |
| **Function** | Hash\_elem e와 동일한 요소를 hash h에서 찾아 제거하고 해당 요소를 return하는 함수. 동일한 요소를 찾을 수 없다면 null을 return함 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | struct hash \*h, hash\_action\_func \*action |
| **Return** | void |
| **Function** | Hash h의 모든 요소들에 대해 action 함수를 수행하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | struct hash\_iterator \*i, struct hash \*h |
| **Return** | void |
| **Function** | Hash h에 대한 iteration을 위한 초기화 함수. Hash iterator i에 첫번째 bucket의 head 주소를 부여함 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i) |
| **Parameter** | struct hash\_iterator \*i |
| **Return** | struct hash\_elem \* |
| **Function** | Hash iteration을 위해 구현된 함수. Hash\_iterator i가 다음 위치를 가리키도록 하고 해당 위치에 대한 hash\_elem을 return함 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | struct hash\_iterator \*i |
| **Return** | struct hash\_elem \* |
| **Function** | 인자로 들어온 hash\_iterator i에 대한 hash\_elem을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | struct hash \*h |
| **Return** | size\_t |
| **Function** | hash h의 요소들의 개수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h) |
| **Parameter** | struct hash \*h |
| **Return** | bool |
| **Function** | hash h의 요소가 하나도 존재하지 않는지에 대한 여부를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size) |
| **Parameter** | const void \*buf\_, size\_t size |
| **Return** | unsigned |
| **Function** | buf의 위치에서 size byte만큼의 데이터에 대한 hash 값을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_) |
| **Parameter** | const char \*s\_ |
| **Return** | unsigned |
| **Function** | 문자열에 대한 hash 값을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i) |
| **Parameter** | int i |
| **Return** | unsigned |
| **Function** | 정수에 대한 hash 값을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static struct list \*find\_bucket (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | struct hash \*h, struct hash\_elem \*e |
| **Return** | struct list \* |
| **Function** | hash\_elem e가 속한 hash의 bucket을 return하는 함수. 찾을 수 없다면 null을 return함 |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \*find\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | struct hash \*h, struct list \*bucket, struct hash\_elem \*e |
| **Return** | struct hash\_elem \* |
| **Function** | hash\_elem e와 동일한 요소가 bucket에 존재하면 해당 요소를 return하는 함수. 없다면 null을 return함 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit (size\_t x) |
| **Parameter** | size\_t x |
| **Return** | size\_t |
| **Function** | x의 least bit를 turn off 하고 x를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t is\_power\_of\_2 (size\_t x) |
| **Parameter** | size\_t x |
| **Return** | size\_t |
| **Function** | x가 2의 제곱수인지 여부를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \*h) |
| **Parameter** | struct hash \*h |
| **Return** | void |
| **Function** | hash h의 hash element 수에 대한 최적의 bucket 수를 계산하여 buckets를 조정하는 함수. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | struct hash \*h, struct list \*bucket, struct hash\_elem \*e |
| **Return** | void |
| **Function** | hash\_elem e를 bucket(list)의 가장 앞자리로 삽입하는 함수 (list\_push\_front) |

|  |  |
| --- | --- |
| **Prototype** | static void  remove\_elem (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | struct hash \*h, struct hash\_elem \*e |
| **Return** | void |
| **Function** | hash h에서 hash\_elem e를 제거하는 함수 (list\_remove) |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx) |
| **Parameter** | size\_t bit\_idx |
| **Return** | size\_t |
| **Function** | bit\_idx에 해당하는 elem\_idx를 return하는 함수. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) |
| **Parameter** | size\_t bit\_idx |
| **Return** | elem\_type |
| **Function** | bit\_idx에 해당하는 위치가 1로 켜진 elem\_type을 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt) |
| **Parameter** | size\_t bit\_cnt |
| **Return** | size\_t |
| **Function** | bit\_cnt 만큼의 bit를 담기 위한 elem의 수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt) |
| **Parameter** | size\_t bit\_cnt |
| **Return** | size\_t |
| **Function** | bit\_cnt 만큼의 bit를 담기 위한 byte 수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b) |
| **Parameter** | const struct bitmap \*b |
| **Return** | elem\_type |
| **Function** | bitmap b의 마지막 elem에서 사용된 bit가 켜진 bit mask를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | size\_t bit\_cnt |
| **Return** | struct bitmap \* |
| **Function** | bit\_cnt 만큼의 bit를 담을 수 있는 bitmap을 생성하여 모든 bit를 false로 설정한 후 return하는 함수. 동적 할당에 실패하면 null을 return함 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size ) |
| **Parameter** | size\_t bit\_cnt, void \*block, size\_t block\_size |
| **Return** | struct bitmap \* |
| **Function** | 미리 할당된 메모리인 block에 bitmap을 create하여 return하는 함수. block\_size는 반드시 bit\_cnt 만큼의 bit를 담을 수 있어야함 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | size\_t bit\_cnt |
| **Return** | size\_t |
| **Function** | bit\_cnt 만큼의 bit를 담기 위한 byte의 수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | struct bitmap \*b |
| **Return** | void |
| **Function** | bitmap b에 대해 할당된 메모리를 해제하여 bitmap을 파괴하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | const struct bitmap \*b |
| **Return** | size\_t |
| **Function** | bitmap b의 bit\_cnt를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | struct bitmap \*b, size\_t idx, bool value |
| **Return** | void |
| **Function** | bitmap b의 idx번째 위치의 bit를 value로 수정하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | struct bitmap \*b, size\_t bit\_idx |
| **Return** | void |
| **Function** | bitmap b의 idx번째 위치의 bit를 1로 설정하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | struct bitmap \*b, size\_t bit\_idx |
| **Return** | void |
| **Function** | bitmap b의 idx번째 위치의 bit를 0으로 설정하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | struct bitmap \*b, size\_t bit\_idx |
| **Return** | void |
| **Function** | bitmap b의 idx번째 위치의 bit를 toggle하는 함수 (0->1, 1->0) |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| **Parameter** | const struct bitmap \*b, size\_t idx |
| **Return** | bool |
| **Function** | bitmap b의 idx번째 위치의 bit에 대한 값을 return하는 함수 (true or false) |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | struct bitmap \*b, bool value |
| **Return** | void |
| **Function** | bitmap b의 모든 bit들을 value로 설정하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | struct bitmap \*b, size\_t start, size\_t cnt, bool value |
| **Return** | void |
| **Function** | bitmap b의 start번째 bit부터 cnt 만큼의 bit를 value로 설정하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | const struct bitmap \*b, size\_t start, size\_t cnt, bool value |
| **Return** | size\_t |
| **Function** | bitmap b의 start번째 bit부터 cnt 만큼의 bit들 중에서 value로 설정된 bit의 개수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | const struct bitmap \*b, size\_t start, size\_t cnt, bool value |
| **Return** | bool |
| **Function** | bitmap b의 start번째 부터 cnt 만큼의 bit들 중에서 value로 설정된 bit가 하나라도 있다면 true, 없다면 false를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | const struct bitmap \*b, size\_t start, size\_t cnt |
| **Return** | bool |
| **Function** | bitmap b의 start번째 부터 cnt 만큼의 bit들 중에서 true로 설정된 bit가 하나라도 있다면 true, 없다면 false를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | const struct bitmap \*b, size\_t start, size\_t cnt |
| **Return** | bool |
| **Function** | bitmap b의 start번째 부터 cnt 만큼의 bit들 중에서 true로 설정된 bit가 하나도 없다면 true, 있다면 false를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | const struct bitmap \*b, size\_t start, size\_t cnt |
| **Return** | bool |
| **Function** | bitmap b의 start번째 부터 cnt 만큼의 bit들이 모두 true로 설정되어있다면 true, 하나라도 false라면 false를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | const struct bitmap \*b, size\_t start, size\_t cnt, bool value |
| **Return** | size\_t |
| **Function** | bitmap b의 start번째 부터 cnt 만큼 연속적으로 value로 설정된 bit들이 있다면 그것의 bit 개수를 return하는 함수. 그런 bit들이 없다면 BITMAP\_ERROR |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | struct bitmap \*b, size\_t start, size\_t cnt, bool value |
| **Return** | size\_t |
| **Function** | bitmap\_scan를 수행하면서 해당 bit들을 value의 반대 값으로 설정하는 함수. Return 값은 bitmap\_scan의 return |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*b) |
| **Parameter** | const struct bitmap \*b |
| **Return** | size\_t |
| **Function** | bitmap b를 file로 저장하기 위해 필요한 byte 수를 return하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | const struct bitmap \*b |
| **Return** | void |
| **Function** | bitmap b의 내용을 16진수로 dump하는 함수 |