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

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 교수님 |
| 학번 : | 20150195 |
| 이름 : | 백인찬 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_search (struct list\* list, size\_t index) |
| **Parameter** | 찾을 list\_elem이 속한 LIST와 INDEX(0부터 시작) |
| **Return** | 찾은 list\_elem |
| **Function** | INDEX 번째에 있는 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\* list, struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | 바꿀 list\_elem A와 B 그리고 두 list\_elem이 속한 LIST |
| **Return** | None. |
| **Function** | LIST내의 두 list\_elem A와 B를 서로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | list\_elem들을 섞을 LIST |
| **Return** | None. |
| **Function** | LIST내의 list\_elem의 순서를 무작위로 섞는다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less\_func2(const struct list\_elem \*a, const struct list\_elem \*b,  void\* aux) |
| **Parameter** | 비교할 두 list\_elem A와 B, 보조인자 aux |
| **Return** | A가 B보다 작으면 true, 크거나 같으면 false |
| **Function** | A와 B가 속한 list\_item의 data의 값을 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less\_func2(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | 비교할 두 hash\_elem A와 B, 보조인자 aux |
| **Return** | A가 B보다 작으면 true, 크거나 같으면 false |
| **Function** | A와 B가 속한 hash\_item의 data의 값을 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem\* e, void\* aux) |
| **Parameter** | hash를 적용할 hash\_elem E, 보조인자 aux |
| **Return** | None. |
| **Function** | E가 속한 hash\_item의 data를 제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple (struct hash\_elem\* e, void\* aux) |
| **Parameter** | hash\_action를 적용할 hash\_elem E, 보조인자 aux |
| **Return** | None. |
| **Function** | E가 속한 hash\_item의 data를 세제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned int hash\_hash\_func2(const struct hash\_elem\* e, void\* aux) |
| **Parameter** | hash\_function을 적용할 hash\_elem E, 보조인자 aux |
| **Return** | E가 속할 bucket의 index |
| **Function** | E가 속한 hash\_item의 data에 대해 hash\_function을 적용하고 그에 따른 bucket index를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | hash를 적용할 I |
| **Return** | I의 hash를 적용한 값 |
| **Function** | I의 첫번째 자리수를 3으로 나눈 나머지 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | 확장할 BITMAP과 그 SIZE |
| **Return** | 성공할 경우 확장된 BITMAP, 실패하면 NULL |
| **Function** | 기존 BITMAP의 크기에 SIZE를 더한 만큼 확장한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dumpdata(struct bitmap \*bitmap) |
| **Parameter** | data를 출력할 BITMAP |
| **Return** | None |
| **Function** | BITMAP의 비트를 하나씩 순회하며 1일 경우 1을 0일 경우 0을 출력한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | list\_entry(LIST\_ELEM, STRUCT, MEMBER) |
| **Parameter** | LIST\_ELEM 이 속한 STRUCT와 STRUCT 내에서 LIST\_ELEM의 멤버변수명 |
| **Return** | STRUCT의 주소 |
| **Function** | LIST\_ELEM이 속한 STRUCT의 주소를 반환하는 매크로 |

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list \* list) |
| **Parameter** | 초기화할 LIST |
| **Return** | None |
| **Function** | LIST의 head와 tail의 prev / next를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_begin(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 head의 next |
| **Function** | LIST의 맨 첫 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_next(struct list\_elem\* elem) |
| **Parameter** | list\_elem ELEM |
| **Return** | ELEM의 다음 원소를 반환한다. |
| **Function** | ELEM의 next를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_end(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 tail |
| **Function** | LIST의 끝인 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_rbegin(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 tail의 prev |
| **Function** | LIST를 역순으로 했을 때 첫 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_prev(struct list\_elem\* elem) |
| **Parameter** | list\_elem ELEM |
| **Return** | ELEM의 prev |
| **Function** | ELEM의 이전 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_rend(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 head |
| **Function** | LIST를 역순으로 했을 때 끝인 head를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_head(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 head |
| **Function** | LIST의 head를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_tail(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 tail |
| **Function** | LIST의 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_tail(struct list\* list) |
| **Parameter** | LIST |
| **Return** | LIST의 tail |
| **Function** | LIST의 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | 삽입할 원소 ELEM과 삽입할 위치에 있는 BEFORE |
| **Return** | None |
| **Function** | BEFORE의 앞에(prev의 위치)에 ELEM을 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | splice할 원소들의 FIRST와 LAST, 삽입할 위치에 있는 BEFORE |
| **Return** | None |
| **Function** | FIRST부터 LAST사이에 있는 원소들을 제거하고 BEFORE의 앞에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 삽입할 원소 ELEM과 ELEM이 속한 LIST |
| **Return** | None |
| **Function** | LIST의 맨 앞에 ELEM을 삽입한다 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*) |
| **Parameter** | 삽입할 원소 ELEM과 ELEM이 속한 LIST |
| **Return** | None |
| **Function** | LIST의 맨 뒤에 ELEM을 삽입한다 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem) |
| **Parameter** | 삭제할 원소 ELEM |
| **Return** | ELEM의 next |
| **Function** | 연결된 리스트 내에서 ELEM을 없앤다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list) |
| **Parameter** | LIST |
| **Return** | LIST의 맨 앞 원소 |
| **Function** | LIST의 맨 앞 원소를 삭제하면서 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list) |
| **Parameter** | LIST |
| **Return** | LIST의 맨 뒤 원소 |
| **Function** | LIST의 맨 뒤 원소를 삭제하면서 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list) |
| **Parameter** | LIST |
| **Return** | LIST의 head의 next |
| **Function** | head를 제외한 LIST의 맨 앞 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list) |
| **Parameter** | LIST |
| **Return** | LIST의 tail의 prev |
| **Function** | tail을 제외한 LIST의 맨 뒤 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | LIST |
| **Return** | LIST의 크기 |
| **Function** | LIST의 원소의 개수 count하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \* list) |
| **Parameter** | LIST |
| **Return** | LIST가 비었으면 true, 아니면 false |
| **Function** | list\_begin(head의 next)과 list\_end(tail)이 같으면 true, 아니면 false |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*list) |
| **Parameter** | LIST |
| **Return** | None |
| **Function** | LIST의 연결된 순서를 역순으로 바꿔준다 |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \* list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 정렬할 LIST와 정렬의 기준이 될 LESS, 보조인자 aux |
| **Return** | None |
| **Function** | LIST내의 list\_elem이 속한 list\_item의 data를 LESS를 기준으로 비교하여 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \* list, struct list\_elem \*elem,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 정렬된 LIST, 적합한 위치에 삽입할 ELEM, 위치를 찾기 위한 기준이 될 LESS, 보조인자 aux |
| **Return** | None |
| **Function** | LIST 내에 맞는 위치를 LESS 기준으로 찾아 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \* list, struct list \*duplicates,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 탐색할 LIST, 삭제된 원소를 넣을 DUPLICATES, 비교 기준이 될 LESS, 보조인자 aux |
| **Return** | None |
| **Function** | LIST 내의 연속된 같은 data를 갖는 원소를 첫번째만 제외하고 삭제하며, DUPLICATES가 null이 아니면 DUPLICATES에 push\_back한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_max(struct list\* list, list\_less\_func\* less, void\* aux) |
| **Parameter** | LIST와 비교 기준의 LESS, 보조인자 aux |
| **Return** | 최대값 원소의 list\_elem |
| **Function** | LIST내의 list\_elem이 속한 list\_item의 data를 비교하여 그 최대값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_min(struct list\* list, list\_less\_func\* less, void\* aux) |
| **Parameter** | LIST와 비교 기준의 LESS, 보조인자 aux |
| **Return** | 최소값 원소의 list\_elem |
| **Function** | LIST내의 list\_elem이 속한 list\_item의 data를 비교하여 그 최소값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | ASSERT(CONDITION) assert(CONDITION) |
| **Parameter** | CONDITION |
| **Return** | None |
| **Function** | CONDITION이 true면 계속 실행하고, false면 프로그램을 중단하는 매크로 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head(struct list\_elem\* elem) |
| **Parameter** | list\_elem ELEM |
| **Return** | ELEM이 특정 리스트의 head이면 true, 아니면 false |
| **Function** | ELEM과 ELEM의 prev, next가 head의 조건과 맞는지 판단한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail(struct list\_elem\* elem) |
| **Parameter** | list\_elem ELEM |
| **Return** | ELEM이 특정 리스트의 tail이면 true, 아니면 false |
| **Function** | ELEM과 ELEM의 prev, next가 tail의 조건과 맞는지 판단한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior(struct list\_elem\* elem) |
| **Parameter** | list\_elem ELEM |
| **Return** | ELEM이 특정 리스트의 head와 tail이 아니면 true, 둘 중 하나라면 false |
| **Function** | ELEM과 ELEM의 prev, next가 head와 tail의 조건과 다른지 판단한다. |

|  |  |
| --- | --- |
| **Prototype** | static void swap(struct list\_elem\*\* a, struct list\_elem\*\* b) |
| **Parameter** | list\_elem의 이중포인터 A와 B |
| **Return** | None |
| **Function** | A와 B가 가리키는 list\_elem의 포인터를 서로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted(struct list\_elem\* a, struct list\_elem\* b,  list\_less\_func\* less, void\* aux) |
| **Parameter** | list\_elem A, B, 비교 기준의 LESS, 보조인자 aux |
| **Return** | 정렬되어 있으면 true, 아니면 false |
| **Function** | list\_elem이 A부터 B까지 정렬되어 있는지 LESS를 기준으로 비교한 후 정렬되어 있으면 true , 아니면 false를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem\* find\_end\_of\_run(struct list\_elem\* a, struct list\_elem\* b, list\_less\_func\* less, void\* aux) |
| **Parameter** | list\_elem A, B, 비교 기준의 LESS, 보조인자 aux |
| **Return** | A부터 B까지 LESS로 비교하여 정렬되지 않은 마지막 원소 |
| **Function** | A부터 B까지 LESS로 비교하여 정렬되지 않은 마지막 원소를 찾아 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** | list\_elem A0, A1B0, B1 과 비교 기준 LESS, 보조인자 aux |
| **Return** | None |
| **Function** | A0부터 A1B0까지의 원소들과 A1B0부터 B1까지의 원소들을 연결한다. A0~A1B0~B0으로 연결되며 LESS를 기준으로 정렬되어야 한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | list\_elem\_to\_hash\_elem(LIST\_ELEM) |
| **Parameter** | LIST\_ELEM |
| **Return** | hash\_elem의 주소 |
| **Function** | list\_entry(LIST\_ELEM, struct hash\_elem, list\_elem)을 응용하여 LIST\_ELEM이 속한 hash\_elem의 주소를 반환하는 매크로 |

|  |  |
| --- | --- |
| **Prototype** | hash\_entry(HASH\_ELEM, STRUCT, MEMBER) |
| **Parameter** | HASH\_ELEM과 그 속한 STRUCT, STRUCT 내에 HASH\_ELEM의 멤버변수 명 |
| **Return** | STRUCT의 주소 |
| **Function** | HASH\_ELEM이 속한 STRUCT의 주소를 반환하는 매크로 |

|  |  |
| --- | --- |
| **Prototype** | static struct list\* find\_bucket(struct hash\* h, struct hash\_elem\* e) |
| **Parameter** | hashtable H와 table에 속한 hash\_elem E |
| **Return** | E가 속한 list인 bucket을 반환 |
| **Function** | H의 hash함수에 E를 적용해 E가 속한 bucket의 index를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem\* find\_elem(struct hash\* h, struct list\* bucket, struct hash\_elem\* e) |
| **Parameter** | hashtable H와 hash\_elem E가 속한 BUCKET |
| **Return** | E와 동일한 hash\_elem을 반환한다. |
| **Function** | BUCKET내에 원소들과 E가 속한 hash\_item의 data를 비교하여 E와 동일한 원소를 찾아 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem(struct hash\* h, struct list\* bucket, struct hash\_elem\* e); |
| **Parameter** | hashtable H와 hash\_elem E가 속한 BUCKET |
| **Return** | None |
| **Function** | BUCKET에 E를 push\_front 한다 |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem(struct hash\* h, struct hash\_elem\* e); |
| **Parameter** | hashtable H와 hash\_elem E |
| **Return** | None |
| **Function** | E를 hashtable H에서 삭제한다 |

|  |  |
| --- | --- |
| **Prototype** | static void rehash(struct hash\* h); |
| **Parameter** | hashtable H |
| **Return** | None |
| **Function** | insert, remove 등으로 변한 상황에 맞게 hashtable H를 다시 구성한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux); |
| **Parameter** | hashtable H, hash\_function hash, 비교의 기준인 함수 less |
| **Return** | 초기화가 정상적으로 진행됐으면 true, 아니면 false |
| **Function** | hashtable H를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor); |
| **Parameter** | hashtable H, hash\_action DESTRUCTOR |
| **Return** | None |
| **Function** | DESTRUCTOR가 null이 아니면 DESTRUCTOR를 이용해 hashtable H의 모든 원소를 없앤다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor); |
| **Parameter** | hashtable H, hash\_action DESTRUCTOR |
| **Return** | None |
| **Function** | hashtable H에 있는 모든 원소를 hash\_clear를 통해 삭제하고, H의 bucket들을 free한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new); |
| **Parameter** | hashtable H, 삽입할 hash\_elem NEW |
| **Return** | 이미 동일한 hash\_elem이 있으면 해당 hash\_elem을 반환, 없으면 null |
| **Function** | NEW에 맞는 bucket을 찾아 삽입한다. 동일한 원소가 있으면 삽입하지 않고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*NEW); |
| **Parameter** | hashtable H, 대체할 hash\_elem NEW |
| **Return** | 대체된 hash\_elem을 반환 |
| **Function** | NEW에 맞는 bucket을 찾고 동일한 원소가 있으면 해당 원소를 삭제한 후 NEW를 삽입한다. 없으면 삭제 없이 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | hashtable H, 찾을 원소 E |
| **Return** | E와 동일한 찾은 원소를 반환 |
| **Function** | H 내에 E의 hash\_item의 data와 동일한 원소를 찾아 반환한다 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | hashtable H, 삭제할 원소 E |
| **Return** | E와 동일한 원소를 반환 |
| **Function** | E의 hash\_item의 data와 동일한 원소를 찾고 삭제한 후 그 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action); |
| **Parameter** | hashtable H, hash\_action ACTION |
| **Return** | None |
| **Function** | H의 모든 원소들에 ACTION을 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h); |
| **Parameter** | hash\_iterator I, hashtable H |
| **Return** | None |
| **Function** | H의 buckets들의 원소를 순회하는 hash\_iterator I를 H의 buckets들 중 맨 첫 원소로 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i); |
| **Parameter** | hash\_iterator I |
| **Return** | i의 다음 hash\_elem을 반환한다. |
| **Function** | I가 순회하고 있는 hash\_elem의 다음 hash\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i); |
| **Parameter** | hash\_iterator I |
| **Return** | I의 현재 hash\_elem을 반환한다. |
| **Function** | I가 현재 순회하고 있는 hash\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h); |
| **Parameter** | hashtable H |
| **Return** | H의 elem\_cnt |
| **Function** | H의 hash\_elem 총 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h); |
| **Parameter** | hashtable H |
| **Return** | H가 비었으면 true, 아니면 false |
| **Function** | H의 elem\_cnt 가 0이면 true, 아니면 false |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size); |
| **Parameter** | hash\_function을 적용할 BUF와 그 SIZE |
| **Return** | BUF의 hash 값 |
| **Function** | BUF의 SIZE byte의 값을 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_); |
| **Parameter** | hash\_function을 적용할 문자열 S |
| **Return** | S의 hash 값 |
| **Function** | S에 hash\_function을 적용한 값을 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i); |
| **Parameter** | hash\_function을 적용할 정수 I |
| **Return** | I의 hash 값 |
| **Function** | hash\_byte함수를 I에 적용해 반환 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | ELEM\_BITS (sizeof (elem\_type) \* CHAR\_BIT) |
| **Parameter** | None |
| **Return** | 32 (bytes) |
| **Function** | ELEM\_BITS를 sizeof(elem\_type) \* CHAR\_BIT의 값으로 정의하는 매크로 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx) |
| **Parameter** | bit 배열의 index인 BIT\_IDX |
| **Return** | elem\_type 배열의 index를 반환 |
| **Function** | bitmap이 내부적으론 elem\_type의 배열이기 때문에 elem\_type의 index를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) |
| **Parameter** | bit 배열의 index인 bit\_idx |
| **Return** | BIT\_IDX가 1인 elem\_type을 반환 |
| **Function** | BIT\_IDX가 1인 elem\_type을 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt) |
| **Parameter** | 총 bit의 개수인 BIT\_CNT |
| **Return** | BIT\_CNT를 위해 필요한 element의 개수를 반환 |
| **Function** | BIT\_CNT를 위해 필요한 element의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt) |
| **Parameter** | 총 bit의 개수인 BIT\_CNT |
| **Return** | BIT\_CNT를 위해 필요한 bytes의 수를 반환 |
| **Function** | BIT\_CNT를 위해 필요한 bytes의 수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b) |
| **Parameter** | bitmap B |
| **Return** | 마지막 비트값을 찾아 계산하여 bit mask값을 반환 |
| **Function** | B의 비트들 중 마지막에 사용된 비트는 1로 set하고 나머지는 0으로 set한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | 총 bit의 개수인 BIT\_CNT |
| **Return** | 생성된 bitmap |
| **Function** | bitmap을 BIT\_CNT만큼 초기화하고 모든 비트값을 false |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size ) |
| **Parameter** | 최소 BIT\_CNT, 메모리 공간 BLOCK, BLOCK\_SIZE |
| **Return** | 생성된 bitmap |
| **Function** | bitmap을 BLOCK내에 정해진 BLOCK\_SIZE만큼 생성한다. BLOCK\_SIZE는 최소 BIT\_CNT여야 한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | 총 bit의 개수인 BIT\_CNT |
| **Return** | BIT\_CNT에 필요한 byte와 bitmap의 size를 더해 반환 |
| **Function** | BIT\_CNT만큼의 bitmap에 필요한 byte를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | free할 bitmap B |
| **Return** | None |
| **Function** | bitmap B의 bits를 free하고 B를 free한다 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | bitmap B |
| **Return** | B의 bit\_cnt |
| **Function** | B의 bits의 총 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | bitmap B, bit index IDX, Boolean VALUE |
| **Return** | None |
| **Function** | IDX에 있는 bit를 VALUE로 set한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bitmap B, bit index IDX |
| **Return** | None |
| **Function** | IDX에 있는 bit를 true(1)로 set한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bitmap B, bit index IDX |
| **Return** | None |
| **Function** | IDX에 있는 bit를 false(0)로 set한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bitmap B, bit index IDX |
| **Return** | None |
| **Function** | IDX에 있는 bit가 true면 false로, false면 true로 set한다 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| **Parameter** | bitmap B, bit index IDX |
| **Return** | IDX의 bit의 value(true/false) |
| **Function** | IDX에 있는 bit의 value를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | bitmap B, Boolean VALUE |
| **Return** | None |
| **Function** | B의 모든 bits를 VALUE로 set한다 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap B, 시작 index START, set할 bit 개수 CNT, Boolean VALUE |
| **Return** | None |
| **Function** | B의 bits 중 START index부터 CNT개의 bits를 VALUE로 set한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap B, 시작 index START, set할 bit 개수 CNT, Boolean VALUE |
| **Return** | VALUE와 일치하는 bits의 개수 |
| **Function** | B의 bits의 START index부터 CNT개의 bits 중 VALUE와 일치하는 bits의 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap B, 시작 index START, set할 bit 개수 CNT, Boolean VALUE |
| **Return** | VALUE와 일치하는 bit의 여부 |
| **Function** | B의 bits의 START index부터 CNT개의 bits 중 VALUE와 일치하는 bit가 하나라도 있으면 true, 없으면 false |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | bitmap B, 시작 index START, set할 bit 개수 CNT, |
| **Return** | true인 bit가 있는지 여부 |
| **Function** | B의 bits의 START index부터 CNT개의 bits 중 true인 bit가 하나라도 있으면 true, 없으면 false |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | bitmap B, 시작 index START, set할 bit 개수 CNT, |
| **Return** | true인 bit가 없는지 여부 |
| **Function** | B의 bits의 START index부터 CNT개의 bits 중 true인 bit가 하나라도 있으면 false, 하나도 없으면 true |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | bitmap B, 시작 index START, set할 bit 개수 CNT, |
| **Return** | 범위 내 모든 원소가 true인지 여부 |
| **Function** | B의 bits의 START index부터 CNT개의 bits 중 모든 bits가 true면 true, 아니면 false |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap B, 시작 index START, bit 개수 CNT, Boolean VALUE |
| **Return** | VALUE와 일치하는 연속된 CNT개의 bit중 첫 bit의 index |
| **Function** | B의 bits의 START index 이후로 연속된 CNT개의 bit이 모두 VALUE로 set되어 있다면 그 연속된 CNT개의 bit 중 첫 bit의 index를 반환, 없으면 BITMAP\_ERROR |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap B, 시작 index START, bit 개수 CNT, Boolean VALUE |
| **Return** | VALUE와 일치하는 연속된 CNT개의 bit중 첫 bit의 index |
| **Function** | B의 bits의 START index 이후로 연속된 CNT개의 bit이 모두 VALUE로 set되어 있다면 그 bits를 모두 !VALUE로 바꾸고, 그 연속된 CNT개의 bit 중 첫 bit의 index를 반환, 없으면 BITMAP\_ERROR |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*b) |
| **Parameter** | bitmap B |
| **Return** | B에 필요한 byte 수 |
| **Function** | 파일에 있는 B를 위해 필요한 byte의 수 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read (struct bitmap \*b, struct file \*file) |
| **Parameter** | 읽어 들일 bitmap B와 B가 있는 FILE |
| **Return** | 성공하면 true, 실패하면 false |
| **Function** | FILE로부터 B를 읽어 들인다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write (const struct bitmap \*b, struct file \*file) |
| **Parameter** | 쓸 bitmap B와 대상이 될 FILE |
| **Return** | 성공하면 true, 실패하면 false |
| **Function** | FILE에 B를 쓴다(저장한다). |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | bitmap B |
| **Return** | None |
| **Function** | B의 bits를 16진수로 화면(stdout)에 출력한다. |