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

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 교수님 |
| 학번 : | 20181618 |
| 이름 : | 김하늘 |
|  |  |

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | **int search\_index(char \*input)** |
| **Parameter** | 입력받은 자료구조의 이름 |
| **Return** | 생성된 자료구조에 대한 인덱스 |
| **Function** | 리스트, 비트맵, 해쉬테이블 전체를 다루는 구조체 배열  struct oobj \*obj[100];  에서 매칭되는 인덱스를 얻는 함수이다.  자료구조이름을 input으로 받고 모든 obj[]의 name 문자열과 비교하며 일치하는 인덱스를 반환한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | **bool list\_less(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux)** |
| **Parameter** | 값을 비교할 list element 2개 |
| **Return** | a < b 일때 true, a>= b일때 false |
| **Function** | list elem에서 list\_entry를 통해 list item에 접근한다. item의 data 값을 비교하여 a가 b보다 작으면 true를 반환하고 a가 b보다 크거나 같으면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | #define list\_entry(LIST\_ELEM, STRUCT, MEMBER) |
| **Parameter** | list item으로 접근하기 위한 list elem, list item의 구조체 타입, elem 타입 |
| **Return** | list item의 주소값 |
| **Function** | list elem 포인터와 매핑되는 list item 포인터를 찾아주는 매크로이다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head (struct list\_elem \*elem) |
| **Parameter** | 확인할 elem의 주소값 |
| **Return** | true or false |
| **Function** | elem이 해당 리스트의 head인지 확인하는 static function. head면 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior (struct list\_elem \*elem) |
| **Parameter** | 확인할 elem의 주소값 |
| **Return** | true or false |
| **Function** | elem이 해당 리스트의 interior인지 확인하는 static function. interior이면 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail (struct list\_elem \*elem) |
| **Parameter** | 확인할 elem의 주소값 |
| **Return** | true or false |
| **Function** | elem이 해당 리스트의 tail인지 확인하는 static function. tail이면 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 없음 |
| **Function** | 동적할당이 완료된 리스트의 초기값을 설정하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 해당 리스트의 첫번째 element |
| **Function** | 리스트의 첫번째 element, 즉 head 다음의 엘리먼트를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*elem) |
| **Parameter** | 리스트의 특정 element |
| **Return** | 입력받은 element의 다음 element |
| **Function** | 해당 리스트에서 입력받은 element의 다음 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 해당 리스트의 마지막 element |
| **Function** | 리스트의 마지막 element, 즉 tail 이전의 엘리먼트를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 해당 리스트의 역방향의 시작 element |
| **Function** | 리스트의 역방향 시작 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*elem) |
| **Parameter** | 리스트의 특정 element |
| **Return** | 입력받은 element의 이전 element |
| **Function** | 해당 리스트에서 입력받은 element의 이전 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 리스트의 head element |
| **Function** | 리스트를 역순으로 반복할때 사용되는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 리스트의 head element |
| **Function** | 리스트를 통해 무언갈 반복하는 행동을 할 때 사용된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_tail (struct list \*list) |
| **Parameter** | 목표 대상 리스트 |
| **Return** | 리스트의 tail element |
| **Function** | 리스트의 tail element를반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | 삽입할 element의 직전 element, 삽입할 element |
| **Return** | 없음 |
| **Function** | 동적할당을 통해 만든 element를 리스트에 추가하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before,  struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | 삽입될 위치의 직전 element, 삽입할 리스트의 시작 element, 직후 element |
| **Return** | 없음 |
| **Function** | 하나의 리스트의 일부 구간을 다른 리스트에 삽입하는 함수이다. first부터 last 직전까지의 elem list를 제거하고, 제거된 리스트를 before elem 뒤에 이어붙인다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 목표 대상 리스트, 삽입할 element |
| **Return** | 없음 |
| **Function** | 리스트의 제일 앞부분에 해당 엘리먼트를 삽입하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 목표 대상 리스트, 삽입할 element |
| **Return** | 없음 |
| **Function** | 리스트의 제일 뒷부분에 해당 엘리먼트를 삽입하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_remove (struct list\_elem \*elem) |
| **Parameter** | 삭제할 element |
| **Return** | 삭제한 element의 다음 element |
| **Function** | 리스트에서 해당 element를 삭제한다. 리스트의 정보가 없어도 삭제 가능하다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | 삭제되는 element |
| **Function** | 리스트에서 제일 첫번째 엘리먼트를 제거 후 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | 삭제되는 element |
| **Function** | 리스트에서 제일 마지막 엘리먼트를 제거 후 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | 리스트의 첫번째 element |
| **Function** | 해당 리스트에서 가장 첫번째 element를 반환한다. 리스트가 비어있을 경우의 행동은 정의되어 있지 않다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_back (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | 리스트의 마지막 element |
| **Function** | 해당 리스트에서 가장 마지막 element를 반환한다. 리스트가 비어있을 경우의 행동은 정의되어 있지 않다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | 리스트의 길이의 size\_t 값 |
| **Function** | 리스트의 길이를 계산하여 size\_t 값으로 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | true of false |
| **Function** | 리스트가 비어있는지 검사하여 비어있을 경우 true를, 아닐경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b) |
| **Parameter** | 값을 바꿀 두 element |
| **Return** | 없음 |
| **Function** | a, b가 가리키는 'struct list\_elem \*' 를 swap해주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*list) |
| **Parameter** | 목표 대상 list |
| **Return** | 없음 |
| **Function** | 리스트의 순서를 반대로 바꾸어주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 값을 비교할 두 element, 비교 함수, 보조 데이터 |
| **Return** | true of false |
| **Function** | 두 엘리먼트의 값을 비교하여 오름차순의 형태와 같은지 확인하는 함수로, a<b이면 true를 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** | 시작 element, 끝 element, 비교함수, 보조데이터 |
| **Return** | 오름차순이 아닌 부분의 끝 element |
| **Function** | 보조 데이터 aux에 따라 오름차순의 run을 찾는다. a~b 사이의 리스트는 연속적이어야한다. |

|  |  |
| --- | --- |
| **Prototype** | static void  inplace\_merge (struct list\_elem \*a0, struct list\_elem \*a1b0,  struct list\_elem \*b1,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 첫번째 범위에 대한 element(a0, a1b0), 두번째 범위에 대한 element(a1b0, b0), 비교함수, 보조데이터 |
| **Return** | 없음 |
| **Function** | element의 a0부터 a1b0, a1b0부터 b0까지 병합하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 목표 대상 list, 비교함수, 보조데이터 |
| **Return** | 없음 |
| **Function** | 입력받은 리스트를 정렬해주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 목표 대상 list, 삽입할 element, 비교 함수, 보조 데이터 |
| **Return** | 없음 |
| **Function** | 정렬되어있는 리스트를 입력받아 새로 삽입할 element를 적절한 위치에 삽입시켜주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*list, struct list \*duplicates,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 목표 대상 list, 이어 붙일 list, 비교 함수, 보조 데이터 |
| **Return** | 없음 |
| **Function** | 목표 대상 list에서 중복된 값을 제거하고, 제거한 값을 duplicates 리스트에 push\_back을 통해 이어붙인다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 목표 대상 list, 비교 함수, 보조 데이터 |
| **Return** | 최대 값을 가진 element |
| **Function** | 해당 리스트에서 가장 큰 값을 가진 element를 찾아 반환한다. 최대값이 여러개일경우 가장 앞에있는 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*  list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 목표대상 list, 비교함수, 보조데이터 |
| **Return** | 최소 값을 가진 element |
| **Function** | 해당 리스트에서 가장 작은 값을 가진 element를 찾아 반환한다. 최소 값이 여러개일경우 가장 앞에있는 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_swap(struct list\_elem \*a, struct list\_elem \*b)** |
| **Parameter** | swap할 두 element |
| **Return** | 없음 |
| **Function** | 리스트에서 list\_item 자체의 위치를 swap해주는 함수이다. a와 b가 두칸 이상 떨어져 있는 경우, a->next 가 b와 같은 경우, a->prev가 b와 같은 경우 세가지로 나누어 처리하였다. |

|  |  |
| --- | --- |
| **Prototype** | **void list\_shuffle(struct list \*list)** |
| **Parameter** | 목표 대상 list |
| **Return** | 없음 |
| **Function** | 리스트 내의 값들을 무작위로 셔플해주는 함수이다. rand 변수를 통해 0번째 element의 값과 그 나머지 element중 한가지와 특정 횟수만큼 swap을 반복하여 셔플을 수행했다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | **unsigned hash\_func(const struct hash\_elem \*e, void \*aux)** |
| **Parameter** | key에 해당하는 element, 보조 데이터 |
| **Return** | 해쉬 함수의 결과 value |
| **Function** | key값을 해쉬 함수에 넣어 얻은 value 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | **bool h\_less\_func(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux)** |
| **Parameter** | 값을 비교할 두 hash element, 보조 데이터 |
| **Return** | true or false |
| **Function** | element와 매핑되는 data를 hash\_entry를 통해 hash item에서 가져오고, a<b일 경우 true를 반환하며 a>=b이면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | #define list\_elem\_to\_hash\_elem(LIST\_ELEM) |
| **Parameter** | 목표 대상 list element |
| **Return** | element와 매핑되는 hash element |
| **Function** | 리스트 element를 hash element로 바꾸어 반환하는 매크로이다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash,  hash\_less\_func \*less, void \*aux) |
| **Parameter** | 목표 대상 hashtable, 해쉬 함수, 비교 함수, 보조 데이터 |
| **Return** | true or false |
| **Function** | hashtable을 초기화하는 함수이다. 정상적으로 초기화 되었을 경우 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 목표 대상 hashtable, 해쉬 삭제 함수 |
| **Return** | 없음 |
| **Function** | hashtable의 모든 element를 제거하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 목표 대상 hashtable, 해쉬 삭제 함수 |
| **Return** | 없음 |
| **Function** | hashtable을 제거하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 목표 대상 hashtable, 삽입할 hash elem |
| **Return** | NULL 또는 입력받은 hash elem |
| **Function** | element를 해쉬 테이블에 삽입하는 함수로, 만약 해당 해쉬테이블에 동일한 값이 없으면 그대로 삽입하고 있을 경우 추가하지 않고 입력받은 element를 다시 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 목표 대상 hashtable, 삽입할 elem |
| **Return** | NULL |
| **Function** | element를 해쉬 테이블에 삽입하는 함수로, 만약 해당 해쉬테이블에 동일한 값이 없으면 그대로 삽입하고 있을 경우 기존에 존재하던 elem는 삭제하고 다시 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 목표 대상 hashtable, 찾을 element |
| **Return** | NULL 또는 찾은 element |
| **Function** | 해당 해쉬테이블에서 파라미터로 입력받은 element와 동일한 element를 찾아 반환한다. 없을 경우 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 목표 대상 hashtable, 삭제할 element |
| **Return** | NULL 또는 삭제한 element |
| **Function** | 해쉬 테이블에서 삭제할 element를 찾고, 테이블에서 삭제 한 뒤 반환한다. 만약 존재하지 않는 element일 경우 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | 목표 대상 hashtalbe, 특수 기능 수행 함수 |
| **Return** | 없음 |
| **Function** | 해당 해쉬테이블에 사용자가 정의한 특수 기능 수행 함수를 적용시킨다. 이번 프로젝트에선 제곱, 세제곱, free의 함수를 정의하였다. |

|  |  |
| --- | --- |
| **Prototype** | **void h\_action\_square(struct hash\_elem \*e, void \*aux)** |
| **Parameter** | 목표 element, 보조 데이터 |
| **Return** | 없음 |
| **Function** | 해당 element의 data 값을 제곱한 값으로 수정해준다. |

|  |  |
| --- | --- |
| **Prototype** | **void h\_action\_triple(struct hash\_elem \*e, void \*aux)** |
| **Parameter** | 목표 element, 보조 데이터 |
| **Return** | 없음 |
| **Function** | 해당 element의 data 값을 세제곱한 값으로 수정해준다. |

|  |  |
| --- | --- |
| **Prototype** | **void h\_action\_free(struct hash\_elem \*e, void \*aux)** |
| **Parameter** | 목표 element, 보조 데이터 |
| **Return** | 없음 |
| **Function** | 해당 element를 free시켜준다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | 탐색, 리턴을 위한 hash iterator, 목표 대상 hashtable |
| **Return** | 없음 |
| **Function** | 해쉬 테이블의 처음 ㅇ리스트의 시작 부분을 반환하느 ㄴ함수이다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_next (struct hash\_iterator \*i) |
| **Parameter** | 탐색, 리턴을 위한 hash iterator |
| **Return** | hash element |
| **Function** | hash의 다음 값을 찾는 함수이다. 값이 없을 경우 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \* hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | 탐색, 리턴을 위한 hash iterator |
| **Return** | hash element |
| **Function** | 현재 hash iterator의 hash element를 찾는 함수이다. 값이 없을 경우 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | 목표 대상 hashtable |
| **Return** | 해쉬 테이블의 크기 |
| **Function** | size\_t의 자료형으로 해당 해쉬 테이블의 크기를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h) |
| **Parameter** | 목표 대상 hashtable |
| **Return** | true of false |
| **Function** | 해당 해쉬테이블이 비어있으면 true, 아니면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size) |
| **Parameter** | key buf, buf의 크기 |
| **Return** | 해쉬 함수의 결과 value |
| **Function** | 해쉬 함수로, key값에 적절한 비트 연산을 수행한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i) |
| **Parameter** | 입력받은 data(key) |
| **Return** | 해쉬 함수의 결과값(value) |
| **Function** | 앞서 정의한 unsigned hash\_bytes()를 통해 value값을 구해 반환한다. 데이터 형은 unsigned이다. |

|  |  |
| --- | --- |
| **Prototype** | **unsigned hash\_int\_2(int i)** |
| **Parameter** | key값 |
| **Return** | value |
| **Function** | 이번 프로젝트의 hashtable의 bucket size는 4이지만 이 해쉬 함수는 bucket의 인덱스 범위는 0~9까지 인것으로 가정했다.  들어오는 값이 음수인 경우도 있었으므로 값을 제곱해주었고, 제곱한 값의 중간의 값을 추출해 해당 값을 return해주는 제곱법을 사용하였다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list \* find\_bucket (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 목표 대상 hashtable, hash element |
| **Return** | bucket |
| **Function** | 해쉬테이블과 element와 매핑되어있는 bucket을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \* find\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 목표 대상 hashtable, bucket, 찾을 elemnet |
| **Return** | hash element |
| **Function** | 해당 버킷에서 element와 같은 element를 찾아 반환한다. 존재하지 않을 경우 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit (size\_t x) |
| **Parameter** | size\_t |
| **Return** | size\_t |
| **Function** | 가장 낮은 순서의 비트가 1로 설정된 상태의 X값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t is\_power\_of\_2 (size\_t x) |
| **Parameter** | size\_t |
| **Return** | true or false |
| **Function** | x가 2의 지수승일 경우 true를 반환하고, 아닐 경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \*h) |
| **Parameter** | 목표 대상 hash table |
| **Return** | 없음 |
| **Function** | 해쉬 테이블의 버킷의 수를 바꾸어주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | static void inserrt\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 목표 대상 hashtable, bucket, 삽입할 element |
| **Return** | 없음 |
| **Function** | 대상 해쉬 테이블의 버킷 list에 element를 추가한다. list\_push\_front를 통해 수행한다. |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 목표 대상 hashtable, bucket, 삭제할 element |
| **Return** | 없음 |
| **Function** | 대상 해쉬 테이블의 버킷 list에 element를 삭제한다. list\_remove를 통해 수행한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | #define ELEM\_BITS (sizeof (elem\_type) \* CHAR\_BIT) |
| **Parameter** | . |
| **Return** | . |
| **Function** | unsigned long의 크기에 char bit크기를 곱해준 값으로, element의 크기를 나타낸다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx) |
| **Parameter** | size\_t인 비트 인덱스 |
| **Return** | size\_t |
| **Function** | bit idx를 포함하는 element의 인덱스를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) |
| **Parameter** | size\_t인 비트 인덱스 |
| **Return** | element |
| **Function** | bit idx가 가리키는 유일한 bit가 1인 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt) |
| **Parameter** | size\_t인 비트 카운트 |
| **Return** | size\_t |
| **Function** | BIT\_CNT에 필요한 element의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt) |
| **Parameter** | size\_t인 비트 카운트 |
| **Return** | size\_t |
| **Function** | BIT\_CNT에 필요한 bytes의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b) |
| **Parameter** | 목표 대상 bitmap |
| **Return** | unsigned long |
| **Function** | 해당 비트맵의 마지막 element는 1로, 나머지는 0으로 설정한 bit mask를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | 비트맵의 크기 |
| **Return** | 초기화한 bitmap |
| **Function** | 비트맵을 초기화하는 함수이다. bit\_cnt 크기의 비트맵을 동적할당을 통해 만들고, 정상적으로 만들어지지 않아씅ㄹ 경우 NULL을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size ) |
| **Parameter** | 비트맵의 크기, 블록, 이를 위한 byte count |
| **Return** | 생성된 비트맵 |
| **Function** | 비트맵을 만드는 함수로, 블록 사이즈가 미리 할당된다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | 비트맵의 bit 수 |
| **Return** | buf의 크기 |
| **Function** | 비트맵의 buf size를 반환하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | 목표 대상 bitmap |
| **Return** | 없음 |
| **Function** | 비트맵을 제거하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | 목표 대상 bitmap |
| **Return** | bitmap의 크기 |
| **Function** | 비트맵의 bits 크기를 반환하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | 목표 대상 bitmap, 해당 인덱스, 원하는 boolean값 |
| **Return** | 없음 |
| **Function** | 입력받은 boolean값으로 bitmap의 해당 인덱스의 값을 바꾸어주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 목표 대상 bitmap, 해당 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 접근해 값을 true로 바꾸는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 목표 대상 bitmap, 해당 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 접근해 값을 false로 바꾸는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 목표 대상 bitmap, 해당 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 접근해 값을 토글하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| **Parameter** | 목표 대상 bitmap, 해당 인덱스 |
| **Return** | true or false |
| **Function** | 해당 비트맵의 인덱스에 저장된 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | 목표 대상 bitmap, 바꿀 boolean 값 |
| **Return** | 없음 |
| **Function** | 해당 비트맵의 모든 값들을 입력받은 boolean 값으로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위, 바꿀 boolean값 |
| **Return** | 없음 |
| **Function** | 비트맵의 start인덱스부터 cnt수 만큼 boolean값으로 바꾸어주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위, 찾을 boolean값 |
| **Return** | 해당하는 개수 |
| **Function** | 비트맵의 start 인덱스부터 cnt개까지 입력받은 boolean값과 일치하는 비트의 수를 계산한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위, 찾을 boolean값 |
| **Return** | true or false |
| **Function** | 해당 비트맵의 주어진 구간에서 한 bit라도 입력받은 boolean값과 일치할 경우 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위 |
| **Return** | true or false |
| **Function** | 해당 비트맵의 주어진 구간에서 한 bit라도 true일 경우 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위 |
| **Return** | true or false |
| **Function** | 해당 비트맵의 주어진 구간에서 한 bit라도 true일 경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위 |
| **Return** | true or false |
| **Function** | 해당 비트맵의 주어진 구간에서 모든 bit가 true일 경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위, boolean |
| **Return** | 연속 구간의 시작 index |
| **Function** | start부터 cnt길이만큼 연속으로 bit드이 입력받은 boolean값으로 set되어 있으면 연속된 구간의 시작 index를 반환하고, 없을 경우 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 목표 대상 bitmap, 시작 인덱스, 원하는 인덱스 범위, boolean |
| **Return** |  |
| **Function** | 해당 구간에서 연속으로 bit들이 입력받은 boolean값으로 set되어 있으면 해당 모든 bit에 값을 flip시켜주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*b) |
| **Parameter** | 목표 대상 bitmap |
| **Return** | byte 수 |
| **Function** | 비트맵에 저장하는데 필요한 byte의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read (struct bitmap \*b, struct file \*file) |
| **Parameter** | 목표 대상 bitmap, file 포인터 |
| **Return** | true or false |
| **Function** | 파일로부터 비트맵을 읽어, 성공적으로 받으면 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write (const struct bitmap \*b, struct file \*file) |
| **Parameter** | 목표 대상 bitmap, file 포인터 |
| **Return** | true or false |
| **Function** | 파일에 비트맵을 쓰는 함수로, 성공적으로 쓰면 true를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | 목표 대상 bitmap |
| **Return** | 없음 |
| **Function** | 비트맵의 내용을 16진수 값으로 dump해주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | **struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size)** |
| **Parameter** | 목표 대상 bitmap, 늘릴 크기 |
| **Return** | 확장된 bitmap |
| **Function** | 입력받은 size만큼 비트맵의 크기를 늘려주는 함수이다. |