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

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 교수님 |
| 학번 : | 20141218 |
| 이름 : | 김찬 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | Void list swap(struct list\_elem \*a, struct list\_item \*b) |
| **Parameter** | Swap할 두 원소 a, b |
| **Return** |  |
| **Function** | 두 원소를 swap한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle (struct list\* list) |
| **Parameter** | Shuffle할 list |
| **Return** |  |
| **Function** | List를 shuffle한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | hash값을 찾을 정수 i |
| **Return** | 정수 i의 hash값을 반환 |
| **Function** | 정수 i의 hash값을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand(struct bitmap\* bitmap, int size) |
| **Parameter** | 확장할 bitmap, 확장할 size |
| **Return** | 확장한 bitmap을 반환 |
| **Function** | Bitmap을 size만큼 더 확장한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less(const struct list\_elem\* a, const struct list\_elem\* b, void\* aux) |
| **Parameter** | 크기를 비교할 두 원소와 보조데이터 aux |
| **Return** | 값을 비교해 뒤 원소가 더 크면 true 아니면 false |
| **Function** | 두 원소의 값을 비교 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem\* a, const struct hash\_elem\* b, void\* aux) |
| **Parameter** | 값을 비교할 hash 원소 2개와 보조데이터 aux |
| **Return** | 뒤 원소가 더 크면 true 아니면 false |
| **Function** | 두 hash 원소의 값을 비교 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem\* a, void\* aux) |
| **Parameter** | 변경할 원소의 주소 |
| **Return** |  |
| **Function** | 해시 값을 두배로 만듬 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem\* a, void\* aux) |
| **Parameter** | 변경할 원소의 주소 |
| **Return** |  |
| **Function** | 해시 값을 세배로 만듬 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destructor(struct hash\_elem\* a, void\* aux) |
| **Parameter** | 삭제할 hash 원소 |
| **Return** |  |
| **Function** | 해당 원소를 free 시킴 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_func(const struct hash\_elem\* a, void\* aux) |
| **Parameter** | key값을 얻을 hash원소 a |
| **Return** | 원소의 key값 |
| **Function** | 원소의 data값을 key로 변환 |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | Static inline bool is\_head (struct list\_elem \*elem) |
| **Parameter** | 첫 노드인지 확인할 list\_elem 구조체 |
| **Return** | Head이면 true 아니면 false |
| **Function** | Elem이 첫 노드인지의 여부를 알려준다. |

|  |  |
| --- | --- |
| **Prototype** | Static inline bool is\_interior (struct list\_elem \*elem) |
| **Parameter** | 내부 노드인지 확인할 list\_elem 구조체 |
| **Return** | Head나 Tail노드가 아니면 true 맞다면 false |
| **Function** | Elem이 Head나 Tai가 아닌 내부 노드인지의 여부를 알려준다. |

|  |  |
| --- | --- |
| **Prototype** | Static inline bool is\_tail (struct list\_elem \*elem) |
| **Parameter** | 끝 노드인지 확인할 list\_elem 구조체 |
| **Return** | Tail이면 true 아니면 false |
| **Function** | Elem이 끝 노드인지의 여부를 알려준다. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_init (struct list \*list) |
| **Parameter** | 초기화할 list 구조체 |
| **Return** |  |
| **Function** | Empty list로 list를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_begin (struct list \*list) |
| **Parameter** | 시작 주소를 찾을 List |
| **Return** | Head에 연결된 첫 노드의 주소 |
| **Function** | List의 시작 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_next (struct list\_elem \*elem) |
| **Parameter** | 다음 list\_elem의 주소를 찾을 elem |
| **Return** | 끝 노드가 아니라면 elem의 다음 주소를 반환 |
| **Function** | elem의 다음 list\_elem 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_end (struct list \*list) |
| **Parameter** | Tail 노드를 찾을 list |
| **Return** | List의 tail |
| **Function** | List의 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_rbegin (struct list \*list) |
| **Parameter** | Reverse beginning 주소를 찾을 list |
| **Return** | Tail에 연결된 첫 노드의 주소 |
| **Function** | List의 reverse beginning 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_prev (struct list\_elem \*elem) |
| **Parameter** | 이전 list\_elem의 주소를 찾을 elem |
| **Return** | 첫 노드가 아니라면 elem의 이전 주소를 반환 |
| **Function** | Elem의 이전 list\_elem 주소를 반환한다. |
| **Prototype** | Struct list\_elem \*list\_rend (struct list \*Iist) |
| **Parameter** | Reverse end 주소를 찾을 list |
| **Return** | List의 Head |
| **Function** | List의 Head 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_head (struct list \*list) |
| **Parameter** | Head 주소를 찾을 list |
| **Return** | List의 Head |
| **Function** | List의 Head 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Struct list\_elem \*list\_tail (struct list \*list) |
| **Parameter** | Tail 주소를 찾을 list |
| **Return** | List의 Tail |
| **Function** | List의 Tail 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_insert(struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | 삽입할 elem과 삽입할 위치를 나타내는 before |
| **Return** |  |
| **Function** | 리스트에 elem를 before앞에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_splice(struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | 삭제할 범위의 first와 last주소, 삽입할 위치를 나타내는 before |
| **Return** |  |
| **Function** | list에서 first부터 last까지 삭제하고 before 앞에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 삽입할 list와 삽입할 elem의 주소 |
| **Return** |  |
| **Function** | Elem를 list의 맨 앞에 push |

|  |  |
| --- | --- |
| **Prototype** | Void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 삽입할 list와 삽입할 elem의 주소 |
| **Return** |  |
| **Function** | Elem를 list의 맨 뒤에 push |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem) |
| **Parameter** | 삭제할 노드의 주소 |
| **Return** | 제거한 노드의 다음 노드 주소 |
| **Function** | list에서 해당 elem를 제거 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list) |
| **Parameter** | 맨 앞 원소를 삭제할 list |
| **Return** | 삭제한 elem의 주소 |
| **Function** | List의 맨 앞 elem를 pop |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list) |
| **Parameter** | 맨 뒤 원소를 삭제할 list |
| **Return** | 삭제한 elem의 주소 |
| **Function** | List의 맨 뒤 elem를 pop |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list) |
| **Parameter** | 맨 앞 원소를 찾을 list |
| **Return** | 맨 앞 elem의 주소 |
| **Function** | List의 맨 앞 elem를 찾는다 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list) |
| **Parameter** | 맨 뒤 원소를 찾을 list |
| **Return** | 맨 뒤 elem의 주소 |
| **Function** | List의 맨 뒤 elem를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | 크기를 찾을 list |
| **Return** | List의 크기 |
| **Function** | List의 크기를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | Bool list\_empty (struct list \*list) |
| **Parameter** | 비어있는지 확인할 list |
| **Return** | 비어있으면 true 아니면 false |
| **Function** | List가 비어있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b) |
| **Parameter** | Swap할 두 elem의 주소 |
| **Return** |  |
| **Function** | 두 elem를 swap한다. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_reverse (struct list \*list) |
| **Parameter** | 순서를 뒤집을 list |
| **Return** |  |
| **Function** | List의 순서를 뒤집는다. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b,  list\_less\_func \*less, void \*aux) |
| **Parameter** | 정렬돼있는지 확인할 구간을 이루는 elem a와 b, 작은 값을 찾는 less function, 보조데이터 aux |
| **Return** | 정렬되있으면 true, 아니면 false |
| **Function** | Elem a부터 b 이전까지 구간이 정렬돼있는지 여부를 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** | 처음으로 정렬되어있지 않은 elem를 찾을 구간을 이루는 elem a와 b, 작은 값을 찾는 less function, 보조데이터 aux |
| **Return** | 처음으로 정렬되어있지 않은 원소를 반환 |
| **Function** | Elem a부터 b이전까지 처음으로 정렬되어있지 않은 원소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static void inplace\_merge (struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux) |
| **Parameter** | 두 구간을 나타내는 elem a0, a1b0, b1과 더 작은 값을 반환하는 less function, 보조데이터 aux |
| **Return** |  |
| **Function** | 정렬되어있는 두 구간 a0~a1b0, a1b0~b1을 정렬되게 합친다. |

|  |  |
| --- | --- |
| **Prototype** | Void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 정렬할 list, 작은 값을 찾는 less function, 보조데이터 aux |
| **Return** |  |
| **Function** | List를 정렬 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| **Parameter** | 삽입할 list와 삽입할 원소 elem, 더 작은 값을 반환하는 less function, 보조데이터 aux |
| **Return** |  |
| **Function** | List에 elem를 정렬 순서에 맞게 삽입 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| **Parameter** | 같은 값이 있는지 확인할 list, 그 값들을 append할 duplicates, 더 작은 값을 반환하는 less function, 보조데이터 aux |
| **Return** |  |
| **Function** | list에서 같은 값들을 list에서 삭제해 duplicates에 append |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 가장 큰 값을 찾을 list와 더 작은 값을 반환하는 less function, 보조데이터 aux |
| **Return** | 가장 큰 elem의 주소 |
| **Function** | list에서 가장 큰 elem를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 가장 작은 값을 찾을 list와 더 작은 값을 반환하는 less function, 보조데이터 aux |
| **Return** | 가장 작은 elem의 주소 |
| **Function** | list에서 가장 작은 elem를 찾는다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux) |
| **Parameter** | 새로 만들 hash table과 hash\_hash\_func, hash\_less\_func |
| **Return** | 초기화에 성공하면 true 아니면 false |
| **Function** | Hash table 초기화 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 삭제할 hash table과 사용할 destructor 함수 |
| **Return** |  |
| **Function** | Hash table의 element와 bucket을 삭제 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 삭제할 hash table과 사용할 destructor 함수 |
| **Return** |  |
| **Function** | Hash table을 삭제 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 삽입할 hash table과 원소 |
| **Return** | 새로운 원소와 같은 값이 없으면 NULL, 있으면 그 element의 주소 |
| **Function** | 새로운 원소를 hash table에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | Hash table과 새로운 원소 |
| **Return** | 값이 이미 있으면 해당 element의 주소, 없으면 NULL |
| **Function** | Hash table에 새 원소를 추가하고 기존의 것이 있으면 대체한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 찾을 hash table과 찾을 원소 |
| **Return** | 값을 찾으면 그 값을, 없으면 NULL |
| **Function** | Hash table안에서 주어진 값을 검색 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 삭제할 hash table과 삭제할 원소 |
| **Return** | 삭제한 원소의 주소 |
| **Function** | 원소를 찾아 hash table에서 삭제 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | Hash table과 적용할 action function |
| **Return** |  |
| **Function** | Hash table에 action function을 적용 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | Hash table과 첫번째 원소를 가르키는 iterator |
| **Return** |  |
| **Function** | 해시테이블의 첫번째 원소를 검색 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i) |
| **Parameter** | 현재 원소의 hash iterator |
| **Return** | Hash table에서 다음 원소의 주소 |
| **Function** | Hash table에서 다음 원소를 검색 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | 현재 hash\_iterator의 주소 |
| **Return** | 현재 element를 반환 마지막이면 NULL |
| **Function** | Hash table에서 현재 element를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | 크기를 확인할 hash table |
| **Return** | Hash table의 크기 |
| **Function** | Hash table의 크기를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h) |
| **Parameter** | 비었는지 확인할 hash table |
| **Return** | Hash table이 비었으면 true, 아니면 false |
| **Function** | Hash table이 비었는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size) |
| **Parameter** | Size를 찾을 buf와, 기준 size |
| **Return** | Hash table의 size bytes의 hash값을 반환 |
| **Function** | Hash table의 size bytes를 hash값 확인 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_) |
| **Parameter** | hash값을 확인할 문자열 |
| **Return** | 문자열의 hash값을 반환 |
| **Function** | 문자열의 hash값을 확인 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i) |
| **Parameter** | hash값을 확인할 정수 i |
| **Return** | 정수 i의 hash값 |
| **Function** | 정수 i의 hash값을 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct list \*find\_bucket (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | Bucket을 찾을 hash table과 bucket을 찾을 원소 |
| **Return** | 해당 원소가 속한 bucket값을 반환 |
| **Function** | Hash table에서 해당 원소가 속한 bucket값을 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \*find\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 원소를 찾을 hash table, 원소를 찾을 bucket, 찾을 원소 |
| **Return** | Hash table의 bucket에서 e와 같은 원소가 있으면 그 주소를 반환 아니면 NULL |
| **Function** | Hash table의 bucket에서 같은 원소가 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit (size\_t x) |
| **Parameter** | Bit를 off할 정수 x |
| **Return** | 최하위비트를 off한 값을 반환 |
| **Function** | 최하위비트를 off해준다. |

|  |  |
| --- | --- |
| **Prototype** | Static inline size\_t is\_power\_of\_2 (size\_t x) |
| **Parameter** | 2의 거듭제곱꼴인지 확인할 정수 x |
| **Return** | 2의 거듭제곱이면 true 아니면 false |
| **Function** | 숫자가 2의제곱꼴인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \*h) |
| **Parameter** | Rehash할 hash table |
| **Return** |  |
| **Function** | Hash table을 rehash 해준다. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 삽입할 hash table, 삽입할 bucket, 삽입할 원소 |
| **Return** |  |
| **Function** | Hash table의 bucket에 원소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 삭제할 hash table, 삭제할 원소 |
| **Return** |  |
| **Function** | Hash table에서 해당 원소를 삭제한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx) |
| **Parameter** | Index를 알고싶은 bit의 number |
| **Return** | 해당 bit의 index값 |
| **Function** | 입력받은 bit\_idx의 index를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) |
| **Parameter** | Bit가 켜져있는지 알고 싶은 bit의 number |
| **Return** | 해당 bit가 켜져있으면 true 아니면 false |
| **Function** | 해당 bit가 켜져있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt) |
| **Parameter** | 해당 bit를 만들기 위해 Element가 얼마나 필요한지 알고싶은 bit number |
| **Return** | Element의 개수를 반환 |
| **Function** | 해당 bit를 만들기 위해 필요한 element의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt) |
| **Parameter** | 해당 bit를 만들기 위해 bytes가 얼마나 필요한지 알고싶은 bit number |
| **Return** | Bytes의 개수를 반환 |
| **Function** | 해당 bit를 만들기 위해 필요한 bytes의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b) |
| **Parameter** | 처음으로 bit가 켜져있는 위치를 찾을 bitmap |
| **Return** | 처음으로 bit가 켜져있는 위치를 반환 |
| **Function** | 처음으로 bit가 켜져있는 위치를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | 원하는 bitmap의 size |
| **Return** | 할당에 성공하면 bitmap의 주소를 아니면 NULL을 반환 |
| **Function** | 입력받은 크기 만큼 bitmap을 할당한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size ) |
| **Parameter** | 원하는 bitmap의 size, 이미 block\_size만큼 할당된 BLOCK |
| **Return** | Block\_size가 Block에 bitmap을 할당 |
| **Function** | 이미 할당돼있는 block에 bitmap을 만든다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | Buf\_size를 측정할 bitmap의 크기 |
| **Return** | Buf\_size를 반환 |
| **Function** | 입력받은 크기만큼 bitmap을 만들 때 필요한 buf\_size |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | 파괴할 bitmap |
| **Return** |  |
| **Function** | Bitmap을 없앤다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | 크기를 측정할 bitmap |
| **Return** | Bitmap의 크기 |
| **Function** | Bitmap의 크기를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | Set할 bitmap, set할 위치의 index, set 값 value |
| **Return** |  |
| **Function** | Bitmap의 index위치의 값을 value로 set한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | Set할 bitmap, true로 설정할 index |
| **Return** |  |
| **Function** | Bitmap의 index위치의 값을 true로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | Set할 bitmap, false로 설정할 index |
| **Return** |  |
| **Function** | Bitmap의 index위치의 값을 false로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | Set할 bitmap, flip할 index |
| **Return** |  |
| **Function** | Bitmap의 index위치의 값이 true면 false로 false면 true로 설정 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| **Parameter** | 값을 찾을 bitmap, 값을 찾을 index |
| **Return** | Bitmap의 index |
| **Function** | Bitmap의 index의 값을 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | Set할 bitmap, set할 value |
| **Return** |  |
| **Function** | Bitmap의 모든 bit의 값을 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Set할 bit map, set을 시작할 위치, set를 할 size, set할 value |
| **Return** |  |
| **Function** | Bitmap의 start부터 cnt만큼 bit값을 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Count할 bitmap, count를 시작할 위치, count할 크기, count할 value |
| **Return** | Bitmap의 start부터 cnt만큼 bit값이 value인 개수 |
| **Function** | Bitmap의 start부터 cnt만큼 bit값이 value인 것의 bit의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Check할 bitmap, check를 시작할 start, check할 크기, check할 value |
| **Return** | Bitmap의 start부터 cnt만큼 bit값이 value인 것이 하나라도 있으면 true 아니면 false |
| **Function** | Bitmap의 start부터 cnt만큼 bit값이 value인 것이 하나라도 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | Check할 bitmap, check를 시작할 start, check할 크기 cnt |
| **Return** | Bitmap의 start부터 cnt만큼 bit값이 하나라도 true이면 true 아니면 false |
| **Function** | Bitmap의 start부터 cnt만큼 bit값이 하나라도 true인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | Check할 bitmap, check를 시작할 start, check할 크기 cnt |
| **Return** | Bitmap의 start부터 cnt만큼 bit값이 하나라도 false이면 true 아니면 false |
| **Function** | Bitmap의 start부터 cnt만큼 bit값이 하나라도 false인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | Check할 bitmap, check를 시작할 start, check할 크기 cnt |
| **Return** |  |
| **Function** | Bitmap의 start부터 cnt만큼 bit값이 모두 true이면 true 아니면 false |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Scan할 bitmap, scan을 시작할 start, scan할 크기 cnt, scan할 value |
| **Return** | Bitmap의 start부터 cnt만큼 value값이 연속으로 존재하면 start를 반환 아니면 Bitmap\_error를 반환 |
| **Function** | Bitmap의 start부터 cnt만큼 해당 value값이 연속으로 존재하는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Scan할 bitmap, scan을 시작할 start, scan할 크기 cnt, scan할 value |
| **Return** | Bitmap의 start부터 cnt만큼 value값이 연속으로 존재하면 start를 반환 아니면 Bitmap\_error를 반환 |
| **Function** | Bitmap의 start부터 cnt만큼 value값이 연속으로 존재하면 start를 반환하고 값을 모두 flip |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | 16진수로 출력할 bitmap |
| **Return** |  |
| **Function** | Bitmap의 값을 16진수로 출력 |