**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 |
| 학번 : | 20201547 |
| 이름 : | 고운 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem\*, struct list\_elem\*) |
| **Parameter** | * struct list\_elem\* elem1 * struct list\_elem\* elem2 |
| **Return** | 없음 |
| **Function** | 두 리스트 요소의 위치를 바꿈. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list\*) |
| **Parameter** | struct list\* list |
| **Return** | 없음 |
| **Function** | 리스트의 요소들을 무작위로 섞음 |

|  |  |
| --- | --- |
| **Prototype** | static bool func\_list\_less(const struct list\_elem\*, const struct list\_elem\*, void\*) |
| **Parameter** | * const struct list\_elem\* elem1 * const struct list\_elem\* elem2: * void\* aux |
| **Return** | Bool |
| **Function** | 두 리스트 요소를 비교하여 작은지 여부를 판단 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* make\_listelem(int); |
| **Parameter** | int data |
| **Return** | 새로운 리스트 요소의 포인터 |
| **Function** | 원하는 값을 가지고 있는 새로운 리스트 요소를 생성 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* make\_hashelem(int) |
| **Parameter** | int data |
| **Return** | 새로운 해시 요소의 포인터 |
| **Function** | 원하는 값을 가지고 있는 새로운 해시 요소를 생성 |

|  |  |
| --- | --- |
| **Prototype** | static unsigned hashfunc(const struct hash\_elem\*, void\*) |
| **Parameter** | * const struct hash\_elem\* h * void\* aux |
| **Return** | 해시값 |
| **Function** | 해시 요소에 대한 해시 값을 계산하고 반환 |

|  |  |
| --- | --- |
| **Prototype** | static bool func\_hash\_less(const struct hash\_elem\*, const struct hash\_elem\*, void\*) |
| **Parameter** | * const struct hash\_elem\* elem1 * const struct hash\_elem\* elem2 * void\* aux |
| **Return** | Bool(비교결과) |
| **Function** | 두 해시 요소를 비교하여 작은지 여부를 판단 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int) |
| **Parameter** | int i |
| **Return** | 해시값 |
| **Function** | 주어진 정수를 해싱하여 해시 값을 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand(struct bitmap\*, int) |
| **Parameter** | * struct bitmap\* bitmap * int size |
| **Return** | 확장된 비트맵의 포인터 |
| **Function** | 비트맵을 주어진 크기만큼 확장하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | void triple\_action(struct hash\_elem\*, void\*) |
| **Parameter** | * struct hash\_elem\* elem * void \* aux |
| **Return** | 없음 |
| **Function** | 헤시 요소의 데이터를 세제곱함. |

|  |  |
| --- | --- |
| **Prototype** | void square\_action(struct hash\_elem\*, void\*) |
| **Parameter** | * struct hash\_elem\* elem: 해시 요소 포인터 * void\* aux: 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | 해시 요소의 데이터를 제곱함. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_begin(struct list\*) |
| **Parameter** | struct list\* list |
| **Return** | 리스트의 첫 번째 요소를 가리키는 포인터 |
| **Function** | 리스트의 첫 번째 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_next(struct list\_elem\*) |
| **Parameter** | struct list\_elem\* elem: 리스트 요소 포인터 |
| **Return** | 다음 요소를 가리키는 포인터 |
| **Function** | 주어진 요소의 다음 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_end(struct list\*) |
| **Parameter** | struct list\_list \*list: 리스트 포인터 |
| **Return** | 리스트의 끝 요소를 나타내는 포인터 |
| **Function** | 리스트의 끝 요소를 나타내는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_rbegin(struct list\*) |
| **Parameter** | struct list \* list: 리스트 포인터 |
| **Return** | 리스트의 역순으로 첫 번째 요소를 가리키는 포인터 |
| **Function** | 리스트의 역순으로 첫 번째 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_prev(struct list\_elem\*) |
| **Parameter** | struct list\_elem\* elem: 리스트 요소 포인터 |
| **Return** | 전 요소를 가리키는 포인터 |
| **Function** | 주어진 요소의 이전 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_rend(struct list\*) |
| **Parameter** | struct list \* list: 리스트 포인터 |
| **Return** | 리스트의 역순으로 끝을 나타내는 포인터 |
| **Function** | 리스트의 역순으로 끝을 나타내는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_head(struct list\*) |
| **Parameter** | struct list \* list: 리스트 포인터 |
| **Return** | 리스트의 head를 나타내는 포인터 |
| **Function** | 라스트의 head를 나타내는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_tail(struct list\*) |
| **Parameter** | struct list \* list: 리스트 포인터 |
| **Return** | 리스트의 tail을 나타내는 포인터 |
| **Function** | 리스트의 tail을 나타내는 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert(struct list\_elem\*, struct list\_elem\*) |
| **Parameter** | struct list\_elem \*before,  struct list\_elem \*elem |
| **Return** | 없음 |
| **Function** | 새 요소를 주어진 위치 이전에 삽입 |

|  |  |
| --- | --- |
| **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** | 없음 |
| **Function** | 현재 리스트에서 지정된 범위의 요소들을 제거한 후, 지정된 위치 이전에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front(struct list\*, struct list\_elem\*) |
| **Parameter** | struct list \*list,  struct list\_elem \*elem |
| **Return** | 없음 |
| **Function** | 리스트의 맨 앞에 새 요소를 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back(struct list\*, struct list\_elem\*) |
| **Parameter** | struct list \*list, struct list\_elem \*elem |
| **Return** | 없음 |
| **Function** | 리스트의 맨 뒤에 새 요소를 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_remove(struct list\_elem\*) |
| **Parameter** | struct list\_elem \*elem |
| **Return** | 제거된 요소의 다음 요소를 가리키는 포인터 |
| **Function** | 리스트에서 주어진 요소를 제거하고, 제거된 요소의 다음 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_pop\_front(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 제거된 리스트의 맨 앞 요소를 가리키는 포인터 |
| **Function** | 리스트의 맨 앞 요소를 제거하고, 제거된 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_pop\_back(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 제거된 리스트의 맨 뒤 요소를 가리키는 포인터 |
| **Function** | 리스트의 맨 뒤 요소를 제거하고, 제거된 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_front(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 리스트의 맨 앞 요소를 가리키는 포인터 |
| **Function** | 리스트의 맨 앞 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_back(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 리스트의 맨 뒤 요소를 가리키는 포인터 |
| **Function** | 리스트의 맨 뒤 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 리스트의 요소 수를 나타내는 값 |
| **Function** | 리스트의 요소 수를 계산하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 리스트가 비어 있으면 true, 비어 있지 않으면 false |
| **Function** | 리스트가 비어 있는지 여부를 확인하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse(struct list\*) |
| **Parameter** | struct list \*list |
| **Return** | 없음 |
| **Function** | 리스트의 순서를 역순으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort(struct list\*,list\_less\_func\*, void\*); |
| **Parameter** | struct list \*list,  list\_less\_func \*less,  void \*aux |
| **Return** | 없음 |
| **Function** | 리스트를 주어진 비교 함수를 사용하여 정렬 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered(struct list\*, struct list\_elem\*,      list\_less\_func\*, void\* aux); |
| **Parameter** | struct list \*list,  struct list\_elem \*elem,  list\_less\_func \*less,  void \*aux |
| **Return** | 없음 |
| **Function** | 정렬된 순서를 유지하면서 리스트에 요소를 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique(struct list\*, struct list\* duplicates,      list\_less\_func\*, void\* aux) |
| **Parameter** | struct list \*list,  struct list \*duplicates,  list\_less\_func \*less,  void \*aux |
| **Return** | 없음 |
| **Function** | 리스트에서 중복되는 요소를 제거하고, 중복된 요소가 있다면 주어진 다른 리스트에 추가 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_max(struct list\*, list\_less\_func\*, void\* aux) |
| **Parameter** | struct list \*list,  list\_less\_func \*less,  void \*aux |
| **Return** | 최대 값을 가진 요소를 가리키는 포인터 |
| **Function** | 리스트에서 가장 큰 값을 가진 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem\* list\_min(struct list\*, list\_less\_func\*, void\* aux) |
| **Parameter** | struct list \*list,  list\_less\_func \*less,  void \*aux |
| **Return** | 최소 값을 가진 요소를 가리키는 포인터 |
| **Function** | 리스트에서 가장 작은 값을 가진 요소를 반환 |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init(struct hash\*, hash\_hash\_func\*, hash\_less\_func\*, void\* aux) |
| **Parameter** | struct hash\* h,  hash\_hash\_func\* hash,  hash\_less\_func\* less,  void\* aux |
| **Return** | True or flase |
| **Function** | 해시 테이블을 초기화하고 필요한 메모리를 할당 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear(struct hash\*, hash\_action\_func\*) |
| **Parameter** | struct hash\* h, hash\_action\_func\* destructor |
| **Return** | 없음 |
| **Function** | 해시 테이블에서 모든 요소를 제거하고, 필요 시 함수를 호출하여 메모리를 해제 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy(struct hash\*, hash\_action\_func\*) |
| **Parameter** | struct hash\* h, hash\_action\_func\* destructor |
| **Return** | 없음 |
| **Function** | 해시 테이블을 없앰 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* hash\_insert(struct hash\*, struct hash\_elem\*) |
| **Parameter** | struct hash\* h, struct hash\_elem\* new |
| **Return** | 삽입된 요소와 동일한 요소 or **NULL** |
| **Function** | 새로운 요소를 해시 테이블에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* hash\_replace(struct hash\*, struct hash\_elem\*) |
| **Parameter** | struct hash\* h,  struct hash\_elem\* new |
| **Return** | 이전에 있던 요소와 동일한 요소 or **NULL** |
| **Function** | 기존 요소를 새로운 요소로 대체 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* hash\_find(struct hash\*, struct hash\_elem\*) |
| **Parameter** | struct hash\* h,  struct hash\_elem\* e |
| **Return** | 찾은 요소의 포인터 or **NULL** |
| **Function** | 해시 테이블에서 주어진 요소와 동일한 요소 찾음. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* hash\_delete(struct hash\*, struct hash\_elem\*) |
| **Parameter** | struct hash\* h,  struct hash\_elem\* e |
| **Return** | 삭제된 요소의 포인터 또는 **NULL** |
| **Function** | 주어진 요소를 해시 테이블에서 제거 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply(struct hash\*, hash\_action\_func\*) |
| **Parameter** | struct hash\* h, hash\_action\_func\* action |
| **Return** | 없음 |
| **Function** | 주어진 함수를 해시 테이블의 각 요소에 적용 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first(struct hash\_iterator\*, struct hash\*) |
| **Parameter** | struct hash\_iterator\* i, struct hash\* h |
| **Return** | 없음 |
| **Function** | 해시 테이블 반복을 위해 iterator를 초기화 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* hash\_next(struct hash\_iterator\*) |
| **Parameter** | struct hash\_iterator\* i |
| **Return** | 다음 요소의 포인터 또는 **NULL** |
| **Function** | 해시 테이블 반복에서 다음 요소로 이동 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem\* hash\_cur(struct hash\_iterator\*) |
| **Parameter** | struct hash\_iterator\* i |
| **Return** | 현재 요소의 포인터 또는 **NULL** |
| **Function** | 해시 테이블에서 현재 요소를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size(struct hash\*) |
| **Parameter** | struct hash\* h |
| **Return** | 해시 테이블의 요소 수 |
| **Function** | 해시 테이블의 요소 수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty(struct hash\*) |
| **Parameter** | struct hash\* h |
| **Return** | 해시 테이블이 비어 있으면 **true**, 그렇지 않으면 **false** |
| **Function** | 해시 테이블이 비어 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes(const void\*, size\_t) |
| **Parameter** | const void\* buf\_,  size\_t size |
| **Return** | 해싱된 결과 값 |
| **Function** | 주어진 데이터 버퍼를 해싱하여 결과를 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string(const char\*) |
| **Parameter** | const char\* s\_ |
| **Return** | 해싱된 결과 값 |
| **Function** | 주어진 문자열을 해싱하여 결과를 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int(int) |
| **Parameter** | int i |
| **Return** | 해싱된 결과 값 |
| **Function** | 주어진 정수를 해싱하여 결과를 반환 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_create(size\_t bit\_cnt) |
| **Parameter** | size\_t bit\_cnt |
| **Return** | 생성된 비트맵의 포인터 또는 **NULL** |
| **Function** | 주어진 비트 수의 비트맵을 생성 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_create\_in\_buf(size\_t bit\_cnt, void\*, size\_t byte\_cnt) |
| **Parameter** | size\_t bit\_cnt, void\*, size\_t byte\_cnt |
| **Return** | 생성된 비트맵의 포인터 |
| **Function** | 주어진 블록에 할당된 크기의 비트맵을 생성 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size(size\_t bit\_cnt) |
| **Parameter** | size\_t bit\_cnt |
| **Return** | 필요한 버퍼 크기 |
| **Function** | 주어진 비트 수에 필요한 저장소 버퍼의 크기를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy(struct bitmap\*) |
| **Parameter** | struct bitmap\* b |
| **Return** | 없음 |
| **Function** | 주어진 비트맵을 소멸하고 사용된 메모리를 해제 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size(const struct bitmap\*) |
| **Parameter** | const struct bitmap\* b |
| **Return** | 비트 수 |
| **Function** | 주어진 비트맵의 비트 수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set(struct bitmap\*, size\_t idx, bool) |
| **Parameter** | struct bitmap\* b, size\_t idx, bool value |
| **Return** | 없음 |
| **Function** | 주어진 비트맵에서 지정된 위치의 비트를 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark(struct bitmap\*, size\_t idx) |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** |  |
| **Function** |  |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset(struct bitmap\*, size\_t idx) |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** | 없음 |
| **Function** | 주어진 비트맵에서 지정된 위치의 비트를 false로 재설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip(struct bitmap\*, size\_t idx) |
| **Parameter** | struct bitmap\* b, size\_t bit\_idx |
| **Return** | 없음 |
| **Function** | 주어진 비트맵에서 지정된 위치의 비트를 토글 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test(const struct bitmap\*, size\_t idx) |
| **Parameter** | const struct bitmap\* b, size\_t idx |
| **Return** | 비트 값 |
| **Function** | 주어진 비트맵에서 지정된 위치의 비트 값을 반환 |

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

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple(struct bitmap\*, size\_t start, size\_t cnt, bool) |
| **Parameter** | struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** |  |
| **Function** |  |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count(const struct bitmap\*, size\_t start, size\_t cnt, bool) |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | 지정된 값으로 설정된 비트의 수 |
| **Function** | 주어진 비트맵에서 지정한 범위 내에서 지정한 값으로 설정된 비트의 수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains(const struct bitmap\*, size\_t start, size\_t cnt, bool) |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | 지정된 값으로 설정된 비트가 포함되어 있는지 여부(true or false) |
| **Function** | 비트맵에서 지정한 범위 내에서 지정한 값으로 설정된 비트가 포함되어 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any(const struct bitmap\*, size\_t start, size\_t cnt) |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt |
| **Return** | bool |
| **Function** | 비트맵에서 지정한 범위 내에서 비트가 하나라도 true로 설정되어 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none(const struct bitmap\*, size\_t start, size\_t cnt) |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt |
| **Return** | bool |
| **Function** | 비트맵에서 지정한 범위 내에서 모든 비트가 false로 설정되어 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all(const struct bitmap\*, size\_t start, size\_t cnt) |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt |
| **Return** | bool |
| **Function** | 어진 비트맵에서 지정된 범위 내에서 모든 비트가 true로 설정되어 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan(const struct bitmap\*, size\_t start, size\_t cnt, bool) |
| **Parameter** | const struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | 지정한 값으로 설정된 비트그룹의 시작 인덱스 |
| **Function** | 지정한 값으로 설정된 연속적인 비트그룹의 시작 인덱스 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip(struct bitmap\*, size\_t start, size\_t cnt, bool) |
| **Parameter** | struct bitmap\* b, size\_t start, size\_t cnt, bool value |
| **Return** | 첫 번째 그룹의 시작 인덱스 or BITMAP\_ERROR |
| **Function** | 비트맵에서 지정한 값으로 설정된 연속적인 비트 그룹을 검색하고 해당 그룹의 모든 비트를 반전 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump(const struct bitmap\*); |
| **Parameter** | const struct bitmap\* b |
| **Return** | 없음 |
| **Function** | 비트맵 내용을 16진수로 덤프. |