**Project #1 : MyLib**

담당 교수: 김영재

학번: 20231632

이름: Jumagul Alua

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void init\_all(void); |
| **Parameter** | 없음 |
| **Return** | 없음(void) |
| **Function** | 모든 데이터 구조(리스트, 해시 테이블, 비크맵)를 초기화 |

|  |  |
| --- | --- |
| **Prototype** | #define hash\_entry(ELEM, STRUCT, MEMBER) |
| **Parameter** | ELEM: 현재 접근 중인 요소의 포인터  STRUCT: 상위 구조체의 타입  MEMBER: 상위 구조체 내부의 멤버 이름 |
| **Return** | STRUCT \*:ELEM이 속한 상위 구조체의 시작 주소 |
| **Function** | ELEM이 속한 상위 구조체의 시작 주소를 계산하여 반환(list\_elem 사용해서 리스트랑 비슷하게 정의) |

|  |  |
| --- | --- |
| **Prototype** | struct foo { struct list\_elem elem; int value; }; |
| **Parameter** | struct list\_elem elem: 리스트에 연결된 요소  int value: 데이터 필드 |
| **Return** | 없음 |
| **Function** | 리스트에서 사용할 대이터 구조 정의 |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_list(struct list \*list); |
| **Parameter** | list: 출력할 리스트의 포인터 |
| **Return** | 없음(void) |
| **Function** | 각 요소 공백으로 구분되며, 리스트의 모든 요소를 출력 |

|  |  |
| --- | --- |
| **Prototype** | void delete\_list(struct list \*list); |
| **Parameter** | list: 삭제할 리스트의 포인터 |
| **Return** | 없음(void) |
| **Function** | 리스트의 모든 요소를 재거하며 메모리를 해제 |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list \*list, int index1, int index2); |
| **Parameter** | list: 요소를 교환할 리스트의 포인터 |
| **Return** | 없음(void) |
| **Function** | 주어진 두 인덱스에 해당하는 요소의 값을 서로 교환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list); |
| **Parameter** | list: 섞을 리스트의 포인터 |
| **Return** | 없음(void) |
| **Function** | Fisher-Yates 셔플 알고리즘을 사용하여 리스트의 요소를 무작위로 섞기 |

|  |  |
| --- | --- |
| **Prototype** | bool less\_func(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux); |
| **Parameter** | a, b: 비교할 두 리스트 요소의 포인터  aux: 추가 데이터(사용 안 함) |
| **Return** | true or false |
| **Function** | 두 요소의 값을 비교하여 a<b인지 체크 |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i); |
| **Parameter** | i: 해시 값을 계산할 정수 |
| **Return** | 입력된 정수 i에 대한 해시 값(insigned) |
| **Function** | 입력된 정수를 다양한 비트 연산과 혼합을 통해 고르게 분포된 해시 값을 생성 |

|  |  |
| --- | --- |
| **Prototype** | unsigned my\_hash\_hash\_func(const struct hash\_elem \*e, void \*aux); |
| **Parameter** | e: 해시 테이블의 요소 포인터  aux: 추가 제니터(사용 안 함) |
| **Return** | 해시 값(unsigned) |
| **Function** | 입력된 요소의 값을 기반으로 해시 값을 계산 |

|  |  |
| --- | --- |
| **Prototype** | bool my\_hash\_less\_func(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux); |
| **Parameter** | a, b: 비교할 두 해시 요소의 포인터  aux: 추가 제니터(사용 안 함) |
| **Return** | true or false |
| **Function** | 두 요소의 값을 비교하여 a<b인지 체크 |

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_hash(struct hash \*h); |
| **Parameter** | h: 출력할 해시 테이블의 포인터 |
| **Return** | 없음(void) |
| **Function** | 각 요소 공백으로 구분되며, 해시 테이블의 모든 요소를 출력 |

|  |  |
| --- | --- |
| **Prototype** | void delete\_hash(struct hash \*h); |
| **Parameter** | h: 삭제할 해시 테이블의 포인터 |
| **Return** | 없음(void) |
| **Function** | 해시 테이블의 모든 요소를 제거하고 초기화 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int extra\_bits) |
| **Parameter** | bitmap: 확장할 비트맵  extra\_bits: 추가할 비트 수 |
| **Return** | 확장된 비트맵 포인터(struct bitmap \*), 실패 시 null |
| **Function** | 기존 비트맵의 크기를 extra\_bits만큼 확장하고, 새러운 비트들은 모두 0으로 초과 |

|  |  |
| --- | --- |
| **Prototype** | int get\_bitmap(const char \*name); |
| **Parameter** | name: 비트맵 이름(예: ‘bm0’, ‘bm1’) |
| **Return** | 비트맵 인덱스(int), 유효하지 않은 이름일 경우 -1 |
| **Function** | 비트맵 이름을 기반으로 해당 비트맵의 인덱스를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_bitmap(struct bitmap \*bm0); |
| **Parameter** | bm0: 출력할 비트맵의 포인터 |
| **Return** | 없음(void) |
| **Function** | 비트맵의 모든 비트 값을 순서대로 출력 |