**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 |
| 학번 : | 20211582 |
| 이름 : | 임나현 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*); |
| **Parameter** | List 자료구조의 pointer |
| **Return** | 없음 |
| **Function** | List의 list\_elem 중에서 2개를 rand함수를 활용하여 랜덤하게 골라 list\_swap시키고, 이를 list\_elem의 개수만큼 반복하여 (for loop) element를 섞는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap (struct list\_elem \*, struct list\_elem \*); |
| **Parameter** | 섞을 두 개의 list\_elem의 pointer |
| **Return** | 없음 |
| **Function** | List의 list\_elem 두 개를 인자로 받아 서로 swap시키는 함수. 해당 element의 prev와 next도 변경하고, element의 prev가 가리키는 next와 element의 next가 가리키는 prev도 변경해야 함. ( 두 element가 연달아 있는 경우, 예외처리를 위해 따로 분리) |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int2 (int); |
| **Parameter** | 해당 정수 |
| **Return** | 해시 값 |
| **Function** | 해당 정수를 해시 함수를 적용한 해시 값을 반환.  이 함수에서는 정수의 각 자리 수를 전부 구해서 합하고, 이를 다시 10으로 나눠 0부터 9까지의 bucket에 들어가게 구현. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size); |
| **Parameter** | Bitmap 자료구조의 포인터, 확장시킬 크기 |
| **Return** | 확장된 bitmap을 가리키는 포인터 |
| **Function** | 인자로 받은 만큼 기존 bitmap의 bit 개수를 뒤로 확장.  기존 bit개수와 확장시킬 크기를 더한 새로운 bitmap를 만들고, 기존 bitmap의 데이터를 bitmap\_mark로 채워서 구현. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hhash\_func(const struct hash\_elem \*e, void \*aux) |
| **Parameter** | Hash\_elem 포인터와 보조 데이터 포인터 |
| **Return** | Hash\_elem의 data의 해시 값 |
| **Function** | Hash\_elem의 data를 hash\_int를 통해 해시 값을 계산하고 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool less\_func(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | Hash\_elem 포인터 두 개와 보조 데이터 포인터 |
| **Return** | a->data 가 b-> data보다 작은 경우 true, 아닌 경우 false |
| **Function** | Hash\_elem 포인터 두 개의 인자가 가리키는 값을 비교하여 작은지 큰지를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void triple\_func (struct hash\_elem \*e, void \*aux) |
| **Parameter** | Hash\_elem 포인터, 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | Hash\_elem 포인터가 가리키는 data 값을 세제곱하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void square\_func (struct hash\_elem \*e, void \*aux) |
| **Parameter** | Hash\_elem 포인터, 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | Hash\_elem 포인터가 가리키는 data 값을 제곱하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void clear\_func (struct hash\_elem \*e, void \*aux) |
| **Parameter** | Hash\_elem 포인터, 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | Hash\_elem 포인터를 삭제하는 함수  Hash\_elem의 이전 element는 다음 element로 hash\_elem의 이후 element를 가리키게 하고,  Hash\_elem의 이후 element는 이전 element로 hash\_elem의 이전 element를 가리키게 구현. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*); |
| **Parameter** | list 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | list 구조체를 빈 리스트로 초기화하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*); |
| **Parameter** | list 구조체의 포인터 |
| **Return** | List의 첫 번째 list\_elem의 포인터 |
| **Function** | List의 head가 가리키고 있는 첫 번째 element의 포인터를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*); |
| **Parameter** | 현재 list\_elem의 포인터 |
| **Return** | 현재 element의 다음 list\_element의 포인터 |
| **Function** | 현재 element에 연결되어 있는 다음 element의 포인터 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | List의 head를 가리키는 포인터 |
| **Function** | List의 head의 포인터 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | List의 tail를 가리키는 포인터 |
| **Function** | List의 head의 포인터 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*, struct list\_elem \*); |
| **Parameter** | 삽입할 위치의 포인터, 삽입할 list\_elem의 포인터 |
| **Return** | 없음 |
| **Function** | 첫 번째 인자로 받은 list\_elem 앞에 두 번째 인자 list\_elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | 삽입할 위치의 포인터, 삽입할 list\_elem의 시작포인터와 끝포인터 |
| **Return** | 없음 |
| **Function** | 첫번째 인자로 받은 list\_elem 앞에 두번째부터 세번째 인자 list\_elem 범위내의 element를 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*); |
| **Parameter** | List 구조체의 포인터, 삽입할 list\_elem의 포인터 |
| **Return** | 없음 |
| **Function** | List의 가장 앞에 새로운 list\_elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*); |
| **Parameter** | List 구조체의 포인터, 삽입할 list\_elem의 포인터 |
| **Return** | 없음 |
| **Function** | List의 가장 뒤에 새로운 list\_elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*); |
| **Parameter** | 제거할 list\_elem의 포인터 |
| **Return** | 제거한 list\_elem의 다음 element를 가리키는 포인터 |
| **Function** | 인자로 받은 list\_elem을 list에서 찾아 제거하고 제거한 element의 다음 element를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | Pop한 list\_elem의 포인터 |
| **Function** | List의 가장 앞에서 list\_elem을 제거 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | Pop한 list\_elem의 포인터 |
| **Function** | List의 가장 뒤에서 list\_elem을 제거 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | List의 크기 |
| **Function** | list에서 element 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | List가 비어 있다면 true, 그렇지 않다면 false |
| **Function** | List가 비어 있는지 여부를 확인하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | List 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | List의 element 순서를 역순으로 뒤집는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*,                  list\_less\_func \*, void \*aux); |
| **Parameter** | List 구조체의 포인터, list\_less function의 포인터, 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | List를 정렬하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*, struct list\_elem \*,                            list\_less\_func \*, void \*aux); |
| **Parameter** | List 구조체의 포인터, list\_less function의 포인터, 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | 새로운 list\_elem을 list에 정렬된 순서에 맞게 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates,                    list\_less\_func \*, void \*aux); |
| **Parameter** | List 구조체의 포인터, 복제할 list의 포인터, list\_less function의 포인터, 보조 데이터 포인터 |
| **Return** | 없음 |
| **Function** | List에서 중복된 요소를 제거하고, 복제할 list가 설정되어 있다면 중복된 요소를 list에 복제 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | List 구조체의 포인터, list\_less function의 포인터, 보조 데이터 포인터 |
| **Return** | list\_elem의 최대값을 가리키는 포인터 |
| **Function** | list에서 list\_item->data가 가장 큰 list\_elem을 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | List 구조체의 포인터, list\_less function의 포인터, 보조 데이터 포인터 |
| **Return** | list\_elem의 최소값을 가리키는 포인터 |
| **Function** | list에서 list\_item->data가 가장 작은 list\_elem을 반환 |

**Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux); |
| **Parameter** | Hash 구조체의 포인터, hash\_hash function의 포인터, hash\_less function의 포인터, 보조 데이터 |
| **Return** | Init을 성공했으면 true, 실패했으면 false |
| **Function** | Hashtable을 초기화 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*, hash\_action\_func \*); |
| **Parameter** | Hash 구조체의 포인터, hash\_action\_function의 포인터 |
| **Return** | 없음 |
| **Function** | Hashtable의 모든 hash\_elem을 제거 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*, hash\_action\_func \*); |
| **Parameter** | Hash 구조체의 포인터, hash\_action\_function의 포인터 |
| **Return** | 없음 |
| **Function** | Hashtable 자체를 파괴 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*); |
| **Parameter** | Hash 구조체의 포인터, 삽입할 hash\_elem의 포인터 |
| **Return** | 이미 같은 element가 있는 경우 해당 element를 반환하고 insert가 된 경우 NULL |
| **Function** | 새로운 hash\_elem을 hashtable에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*); |
| **Parameter** | Hash 구조체의 포인터, 찾을 hash\_elem의 포인터 |
| **Return** | 찾았다면 hash\_elem의 포인터 반환, 없으면 NULL |
| **Function** | hashtable에서 인자로 받은 hash\_elem을 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*, hash\_action\_func \*); |
| **Parameter** | Hash 구조체의 포인터, hash\_action function의 포인터 |
| **Return** | 없음 |
| **Function** | Hashtable의 모든 hash\_elem의 data에 function을 적용하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*); |
| **Parameter** | Hash 자료구조의 포인터 |
| **Return** | Hash의 크기 |
| **Function** | Hashtable의 hash\_element의 개수 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*); |
| **Parameter** | Hash 자료구조의 포인터 |
| **Return** | Hash가 비어 있다면 true, 그렇지 않다면 false |
| **Function** | Hashtable에 hash\_element가 아무것도 없는지 여부를 확인하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int); |
| **Parameter** | 해당 정수 |
| **Return** | 해시 값 |
| **Function** | 해당 정수에 해당하는 해시 함수를 적용한 해시 값을 반환 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | Bitmap의 bit 개수 |
| **Return** | 생성한 bitmap의 포인터 |
| **Function** | 인자로 받은 bit개수를 가진 bitmap을 생성 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*); |
| **Parameter** | Bitmap 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | Bitmap을 제거하고 할당한 메모리를 free해주는 함수 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*); |
| **Parameter** | Bitmap 구조체의 포인터 |
| **Return** | Bitmap의 크기 |
| **Function** | Bitmap의 크기를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 해당 인덱스, 설정할 값 |
| **Return** | 없음 |
| **Function** | Bitmap의 해당 인덱스를 설정 값으로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap 구조체의 포인터, 해당 인덱스 |
| **Return** | 없음 |
| **Function** | Bitmap의 해당 인덱스를 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap 구조체의 포인터, 해당 인덱스 |
| **Return** | 없음 |
| **Function** | Bitmap의 해당 인덱스의 값을 뒤집어주는 함수 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap 구조체의 포인터, 해당 인덱스 |
| **Return** | Bitmap의 해당 인덱스의 값 |
| **Function** | Bitmap의 해당 인덱스의 값 ( 0: false, 1: true) 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 설정할 값 |
| **Return** | 없음 |
| **Function** | Bitmap의 모든 bit의 값을 설정 값으로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스, 인덱스 개수, 설정할 값 |
| **Return** | 없음 |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내의 bit값을 설정 값으로 설정 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스, 인덱스 개수, 비교할 값 |
| **Return** | 일치하는 비트의 개수 |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내의 bit값이 비교 값과 같은 bit의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스. 인덱스 개수, 비교할 값 |
| **Return** | 비교 값과 같은 비트가 있으면 true, 없으면 false |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내의 bit 값이 비교 값과 같은지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스. 인덱스 개수 |
| **Return** | 설정된 비트가 한 개 이상 있으면 true, 없으면 false |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내의 설정된 bit 값이 한 개 이상 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스. 인덱스 개수 |
| **Return** | 설정된 비트가 하나도 없으면 true, 없으면 false |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내의 설정된 bit 값이 한 개도 없는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스. 인덱스 개수 |
| **Return** | 범위 내의 bit가 모두 설정되어 있으면 true, 없으면 false |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내의 bit 값이 모두 설정되어 있는지 여부를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스. 인덱스 개수, 비교할 값 |
| **Return** | 비교 값과 같은 값을 가진 bit group의 시작 인덱스 |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내에서 비교 값과 일치하는 bit group의 첫 번째 인덱스를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | Bitmap 구조체의 포인터, 시작 인덱스. 인덱스 개수, 비교할 값 |
| **Return** | 비교 값과 같은 값을 가진 bit group의 시작 인덱스 |
| **Function** | Bitmap의 시작 인덱스부터 cnt개의 인덱스 범위 내에서 비교 값과 일치하는 bit group을 flip하고 첫 번째 인덱스를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*); |
| **Parameter** | Bitmap 구조체의 포인터 |
| **Return** | 없음 |
| **Function** | Bitmap을 16진수로 출력하는 함수 |