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

|  |  |
| --- | --- |
| 담당 교수 : | 문의현 교수님 |
| 학번 : | 20181255 |
| 이름 : | 김기철 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | elem을 바꿔줄 a, b |
| **Return** | 없음 |
| **Function** | a와 b를 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_nearswap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | 바로 옆의 elem을 바꿔줄 a, b |
| **Return** | 없음 |
| **Function** | 인접한 a와 b를 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | 무작위로 셔플해줄 list |
| **Return** | 없음 |
| **Function** | list를 무작위로 셔플해준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_destroy(struct list \*list) |
| **Parameter** | free할 list |
| **Return** | 없음 |
| **Function** | list를 free한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_cmp(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | 비교할 원소 a와 b, aux |
| **Return** | a의 데이터<b의 데이터이면 1 반환 |
| **Function** | a의 데이터와 b의 데이터를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_func(const struct hash\_elem \*e, void \*aux) |
| **Parameter** | 원소 e, aux |
| **Return** | e의 데이터값을 바꾸어 반환한다. |
| **Function** | hash 구조체에 들어가는 hash 함수로 원소의 값을 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_cmp(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | 비교할 원소 a와 b, aux |
| **Return** | a의 데이터<b의 데이터이면 1 반환 |
| **Function** | a의 데이터와 b의 데이터를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 원소 e, aux |
| **Return** | 없음 |
| **Function** | 원소 e의 데이터를 제곱해준다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 원소 e, aux |
| **Return** | 없음 |
| **Function** | 원소 e의 데이터를 세제곱해준다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destructor(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 원소 e, aux |
| **Return** | 없음 |
| **Function** | 원소 e를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | 값을 바꿔줄 i |
| **Return** | 값이 바뀐 i |
| **Function** | hash\_bytes 함수를 통해 i의 값이 바뀐다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | 길이를 늘려준 bitmap, 늘려줄 길이 size |
| **Return** | 길이를 늘려준 bitmap |
| **Function** | size만큼 bitmap의 길이를 늘려준다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list \*list) |
| **Parameter** | 초기화할 list |
| **Return** | 없음 |
| **Function** | list를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty(struct list \*list) |
| **Parameter** | 비어있는지 확인할 list |
| **Return** | 비었으면 true, 그렇지 않으면 false |
| **Function** | list가 비어있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin(struct list \*list) |
| **Parameter** | 첫번째 list\_elem을 찾고 싶은 list |
| **Return** | 첫번째 list\_elem |
| **Function** | list의 첫번째 list\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next(struct list\_elem \*elem) |
| **Parameter** | 다음 list\_elem을 찾고 싶은 elem |
| **Return** | 다음 list\_elem |
| **Function** | 다음 list\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end(struct list \*list) |
| **Parameter** | 마지막 list\_elem을 찾고 싶은 list |
| **Return** | 마지막 list\_elem |
| **Function** | list의 마지막 list\_elem을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front(struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 새로운 elem과 이것을 넣어줄 list |
| **Return** | 없음 |
| **Function** | list의 맨 앞에 새로운 list\_elem을 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back(struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 새로운 elem과 이것을 넣어줄 list |
| **Return** | 없음 |
| **Function** | list의 맨 뒤에 새로운 list\_elem을 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert(struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | 새로운 elem과 이것을 넣어줄 before |
| **Return** | 없음 |
| **Function** | before 전에 새로운 elem을 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered(struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| **Parameter** | list, 새로 넣어줄 elem, 비교함수 less, aux |
| **Return** | 없음 |
| **Function** | list의 알맞은 위치에 새로운 elem을 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice(struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | 자르는 위치 before, 잘라서 넣어줄 first부터 last |
| **Return** | 없음 |
| **Function** | before 전에 자르고 first부터 last 전까지 잘라서 붙인다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front(struct list \*list) |
| **Parameter** | elem을 삭제할 list |
| **Return** | 삭제한 list\_elem |
| **Function** | list에서 맨 앞의 elem을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back(struct list \*list) |
| **Parameter** | elem을 삭제할 list |
| **Return** | 삭제한 list\_elem |
| **Function** | list에서 맨 뒤의 elem을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove(struct list\_elem \*elem) |
| **Parameter** | 삭제할 elem |
| **Return** | 삭제한 elem |
| **Function** | elem을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front(struct list \*list) |
| **Parameter** | 맨 앞의 elem을 찾고 싶은 list |
| **Return** | 맨 앞의 elem |
| **Function** | list의 맨 앞의 elem을 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back(struct list \*list) |
| **Parameter** | 맨 뒤의 elem을 찾고 싶은 list |
| **Return** | 맨 뒤의 elem |
| **Function** | list의 맨 뒤의 elem을 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size(struct list \*list) |
| **Parameter** | 크기를 찾고싶은 list |
| **Return** | list의 크기 |
| **Function** | list의 크기를 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max(struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 최댓값을 찾고싶은 list, 비교함수 less, aux |
| **Return** | 최댓값 elem |
| **Function** | list의 최댓값을 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min(struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 최솟값을 찾고싶은 list, 비교함수 less, aux |
| **Return** | 최솟값 elem |
| **Function** | list의 최솟값을 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse(struct list \*list) |
| **Parameter** | 역순으로 정렬할 list |
| **Return** | 없음 |
| **Function** | list를 역순으로 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort(struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 정렬할 list, 비교함수 less, aux |
| **Return** | 없음 |
| **Function** | list를 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique(struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| **Parameter** | 중복제거할 list, 중복원소를 담을 duplicates, 비교함수 less, aux |
| **Return** | 없음 |
| **Function** | list의 중복원소를 제거하여 duplicates에 담는다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init(struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux) |
| **Parameter** | 초기화할 hash h, hash함수 hash, 비교함수 less, aux |
| **Return** | 제대로 생성되면 true, 아니면 false 반환 |
| **Function** | hash를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy(struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 삭제할 hash h, 삭제함수 destructor |
| **Return** | 없음 |
| **Function** | hash를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty(struct hash \*h) |
| **Parameter** | 비어있는 hash인지 확인할 h |
| **Return** | 비어있으면 true, 아니면 false 반환 |
| **Function** | hash가 비어있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first(struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | 탐색할 hash h와 iterator i |
| **Return** | 없음 |
| **Function** | i가 h의 첫 번째 원소를 가르키게 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next(struct hash\_iterator \*i) |
| **Parameter** | iterator i |
| **Return** | hash의 다음 원소 |
| **Function** | i가 다음 원소를 가르키게 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert(struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | 새로 넣어줄 원소 new와 hash h |
| **Return** | new와 같은 것이 있으면 반환, 없으면 NULL 반환 |
| **Function** | 새로운 원소 new를 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete(struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 삭제할 원소 e와 hash h |
| **Return** | e와 같은 것이 있으면 반환, 없으면 NULL 반환 |
| **Function** | 원소 e를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size(struct hash \*h) |
| **Parameter** | 크기를 찾고싶은 hash h |
| **Return** | hash h의 크기 |
| **Function** | hash h의 크기를 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear(struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | clear할 hash h, 삭제함수 destructor |
| **Return** | 없음 |
| **Function** | h를 clear한다. (모두 삭제한다.) |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply(struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | 적용할 hash 함수 action, hash h |
| **Return** | 없음 |
| **Function** | action 함수에 따라 h가 변한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find(struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | hash h와 찾고싶은 원소 e |
| **Return** | 찾으면 elem, 못찾으면 NULL 반환 |
| **Function** | h에 원소 e를 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace(struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | replace해줄 hash h, 새로운 원소 new |
| **Return** | replace해주면 원소, 아니면 NULL 반환 |
| **Function** | h에 new를 replace한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int(int i) |
| **Parameter** | 값을 바꿔줄 i |
| **Return** | 값이 바뀐 i |
| **Function** | hash\_bytes 함수를 통해 i의 값이 바뀐다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create(size\_t bit\_cnt) |
| **Parameter** | 새로 만들 bitmap의 크기 cnt |
| **Return** | 새로 만든 bitmap |
| **Function** | cnt 크기의 bitmap을 새로 만든다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy(struct bitmap \*b) |
| **Parameter** | 삭제할 bitmap b |
| **Return** | 없음 |
| **Function** | bitmap을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size(const struct bitmap \*b) |
| **Parameter** | 크기를 찾고 싶은 bitmap b |
| **Return** | bitmap b의 크기 |
| **Function** | bitmap b의 크기를 찾아준다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test(const struct bitmap \*b, size\_t idx) |
| **Parameter** | bitmap b, 1인지 확인할 인덱스 idx |
| **Return** | 1이면 true, 0이면 false 반환 |
| **Function** | idx번째 값이 1인지 확인해준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark(struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bitmap b, 1로 바꿔줄 인덱스 idx |
| **Return** | 없음 |
| **Function** | idx번째 값을 1로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains(const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt, bool형 value |
| **Return** | 1이면 true, 0이면 false 반환 |
| **Function** | start부터 cnt개의 값이 모두 value인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all(const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt |
| **Return** | 1이면 true, 0이면 false 반환 |
| **Function** | start부터 cnt개의 값이 모두 1인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any(const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt |
| **Return** | 1이면 true, 0이면 false 반환 |
| **Function** | start부터 cnt개의 값 중 하나라도 1인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none(const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt |
| **Return** | 1이면 true, 0이면 false 반환 |
| **Function** | start부터 cnt개의 값이 모두 0인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count(const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt, bool형 value |
| **Return** | 1의 갯수 반환 |
| **Function** | start부터 cnt개의 값 중 1의 갯수를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump(const struct bitmap \*b) |
| **Parameter** | 16진수로 바꿀 bitmap b |
| **Return** | 없음 |
| **Function** | bitmap의 값을 16진수로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set(struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | bitmap b, 인덱스 idx, bool형 value |
| **Return** | 없음 |
| **Function** | idx번째 값을 value로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset(struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bitmap b, 인덱스 idx |
| **Return** | 없음 |
| **Function** | bitmap의 값을 모두 0으로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple(struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt, bool형 value |
| **Return** | 없음 |
| **Function** | start부터 cnt개의 값을 value로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all(struct bitmap \*b, bool value) |
| **Parameter** | bitmap b, bool형 value |
| **Return** | 없음 |
| **Function** | bitmap의 값을 모두 value로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan(const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt, bool형 value |
| **Return** | start부터 cnt개의 갯수만큼 모두 value로 이루어진 시작 인덱스를 반환 |
| **Function** | start부터 cnt개의 갯수만큼 모두 value로 이루어져있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip(struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bitmap b, 인덱스 idx |
| **Return** | 없음 |
| **Function** | idx번째 값을 0이면 1로, 1이면 0으로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip(struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | bitmap b, 시작 인덱스 start, 갯수 cnt, bool형 value |
| **Return** | start부터 cnt개의 갯수만큼 모두 value로 이루어진 시작 인덱스를 반환 |
| **Function** | start부터 cnt개의 갯수만큼 모두 value로 이루어져 있으면 0을 1로, 1을 0으로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size(const struct bitmap \*b) |
| **Parameter** | 크기를 알고 싶은 bitmap b |
| **Return** | bitmap의 크기를 반환 |
| **Function** | bitmap의 크기를 찾아준다. |