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

|  |  |
| --- | --- |
| 담당 교수 : | 문의현 교수님 |
| 학번 : | 20191112 |
| 이름 : | 심은서 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | 서로 swap될 두개의 list elements |
| **Return** | 없음 |
| **Function** | 두개의 list elements를 서로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | 무작위로 값이 shuffle될 list |
| **Return** | 없음 |
| **Function** | lis의 elements를 무작위로 뒤섞는다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less(const struct list\_elem\* a, struct list\_elem\* b, void\* aux) |
| **Parameter** | 비교하려는 두개의 list elements |
| **Return** | a의 data<b의 data이면 true, 그렇지 않으면 false |
| **Function** | a의 data와 b의 data를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_hash(const struct hash\_elem\* e, void\* aux) |
| **Parameter** | hash될 hash element |
| **Return** | e의 hash 결과값 |
| **Function** | 특정 hash value에 data를 맵핑하기 위하여 hash\_int를 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem\* a, struct hash\_elem\* b, void\* aux) |
| **Parameter** | 비교하려는 두개의 hash elements |
| **Return** | a의 data<b의 data이면 true, 그렇지 않으면 false |
| **Function** | a의 data와 b의 data를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem\* e, void\* aux) |
| **Parameter** | 제곱할 hash element |
| **Return** | 없음 |
| **Function** | e의 data를 제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem\* e, void\* aux) |
| **Parameter** | 세제곱할 hash element |
| **Return** | 없음 |
| **Function** | e의 data를 세제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | hash\_free(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 메모리를 해제할 hash element |
| **Return** | 없음 |
| **Function** | e의 메모리를 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int i) |
| **Parameter** | hash될 정수 |
| **Return** | i의 hash 값 |
| **Function** | 특정 hash value에 data를 맵핑한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | 크기를 늘리려는 bitmap |
| **Return** | 크기가 확장된 bitmap |
| **Function** | size만큼 bitmap의 크기를 확장한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list\*) |
| **Parameter** | 초기화 될 list |
| **Return** | 없음 |
| **Function** | list를 초기화시킨다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin(struct list \*) |
| **Parameter** | 첫 element에 접근할 list |
| **Return** | list의 첫번째 element |
| **Function** | list의 맨 첫번째 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next(struct list\_elem \*elem) |
| **Parameter** | next element에 접근할 list element |
| **Return** | list element의 다음 element |
| **Function** | list elemet의 다음에 있는 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end(struct list \*) |
| **Parameter** | 마지막 element에 접근할 list |
| **Return** | list의 마지막 element |
| **Function** | list의 맨 마지막 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev(struct list\_elem \*) |
| **Parameter** | 이전 element에 접근할 list element |
| **Return** | list element의 이전에 있는 element |
| **Function** | list element의 이전에 있는 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list) |
| **Parameter** | head에 접근할 list |
| **Return** | list의 head |
| **Function** | list의 head를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*list) |
| **Parameter** | tail에 접근할 list |
| **Return** | list의 tail |
| **Function** | list의 tail을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | list element를 삽입할 위치, 삽입할 list element |
| **Return** | 없음 |
| **Function** | before의 위치에 elem을 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | list element를 삽입할 위치, 삽입할 목록의 첫번째 element, 삽입할 목록의 마지막 element |
| **Return** | 없음 |
| **Function** | first부터 last까지의 element를 기존 list에서 삭제하고 before위치에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 삽입될 list element, list element가 맨 앞에 삽입될 list |
| **Return** | 없음 |
| **Function** | list의 맨 앞에 elem을 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| **Parameter** | 삽입될 list elemet, list element가 맨 뒤에 삽입될 list |
| **Return** | 없음 |
| **Function** | list의 맨 뒤에 elem을 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem) |
| **Parameter** | 삭제될 list element |
| **Return** | 삭제된 list element의 다음 element |
| **Function** | list에서 list element를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list) |
| **Parameter** | 맨 앞 list element를 삭제할 list |
| **Return** | 삭제된 맨 앞 list element |
| **Function** | list의 맨 앞 list element를 삭제하고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list) |
| **Parameter** | 맨 뒤 list element를 삭제할 list |
| **Return** | 삭제된 맨 뒤 list element |
| **Function** | list의 맨 뒤 list element를 삭제하고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list) |
| **Parameter** | 맨 앞 list element를 찾아낼 list |
| **Return** | 맨 앞 list element |
| **Function** | list의 맨 앞 list element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list) |
| **Parameter** | 맨 뒤 list element를 찾아낼 list |
| **Return** | 맨 뒤 list element |
| **Function** | list의 맨 뒤 list element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list) |
| **Parameter** | 크기를 알아낼 list |
| **Return** | list의 elements 수 |
| **Function** | list에 있는 elements의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*list) |
| **Parameter** | 비어있는지 확인할 list |
| **Return** | list가 비어있으면 true, 아니면 false |
| **Function** | list가 비어있는지 여부를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*list) |
| **Parameter** | 순서를 뒤바꿀 list |
| **Return** | 없음 |
| **Function** | list의 순서를 뒤집는다. |

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

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| **Parameter** | element를 삽입할 list, 삽입될 element, 정렬에 사용할 list\_less\_func |
| **Return** | 없음 |
| **Function** | list\_less\_func에 따라서 list의 적절한 위치에 element를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| **Parameter** | 중복된 element가 삭제될 list, 삭제된 element를 삽입할 list, list\_less\_func |
| **Return** | 없음 |
| **Function** | 중복되는 값이 없도록 list\_less\_func에 따라서 list element를 적절히 삭제하고 삭제된 element들을 duplicates에 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 최대값을 찾을 list, 최대값을 찾기 위한 list\_less\_func |
| **Return** | list에서 가장 큰 값을 가진 list element |
| **Function** | list\_less\_func에 따라서 list에서 가장 큰 값을 가진 list element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| **Parameter** | 최소값을 찾을 list, 최소값을 찾기 위한 list\_less\_func |
| **Return** | list에서 가장 작은 값을 가진 list element |
| **Function** | list\_less\_func에 따라서 list에서 가장 작은 값을 가진 list element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | #define list\_entry (list\_elem, struct, member) |
| **Parameter** | list element, list elem이 들어있는 struct, list elem의 멤버명 |
| **Return** | struct |
| **Function** | pointer로 list element가 들어있는 struct에 접근할수 있도록 한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | void hash\_init (struct hash \*h, hash\_hash\_func \*hash,  hash\_less\_func \*less, void \*aux) |
| **Parameter** | 초기화 할 hash, hash 함수, 값을 비교하기 위한 hash\_less\_func |
| **Return** | 없음 |
| **Function** | h를 초기화하고 hash\_hash\_func과 hash\_less\_func을 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | hash action 함수를 적용할 hash, 적용할 hash\_action\_func |
| **Return** | 없음 |
| **Function** | h의 각 element에 대해 hash action 함수를 임의 순서로 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 모든 elements를 삭제할 hash, 삭제에 사용될 hash\_action\_func |
| **Return** | 없음 |
| **Function** | h의 모든 elements를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | 해제할 hash, 해제에 사용되는 hash\_action\_func |
| **Return** | 없음 |
| **Function** | h를 메모리 해제한다 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | element를 삽입할 hash, 삽입될 hash element |
| **Return** | 같은 element가 이미 존재한다면 그 element를 반환, 그렇지 않으면 NULL을 반환한다. |
| **Function** | h에 hash element를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | element가 대체될 hash, 대체될 hash element |
| **Return** | 같은 element가 이미 존재한다면 그 element를 반환, 그렇지 않으면 NULL을 반환한다. |
| **Function** | h에 존재하는 element를 대체하여 새로운 element를 삽입하고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 탐색할 hash, 찾을 hash element |
| **Return** | 같은 element가 이미 존재한다면 그 element를 반환, 그렇지 않으면 NULL을 반환한다. |
| **Function** | h에서 e와 동일한 element를 찾아낸다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | element를 삭제할 hash, 삭제될 element |
| **Return** | 같은 element가 이미 존재한다면 그 element를 반환, 그렇지 않으면 NULL을 반환한다. |
| **Function** | h에서 e를 찾아서 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | iterator i, 첫번째 element를 찾아낼 hash |
| **Return** | 없음 |
| **Function** | hash의 첫번째 element 앞의 iterator를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i) |
| **Parameter** | iterator i |
| **Return** | 다음 element, 없다면 NULL |
| **Function** | i를 다음 element로 이동시키고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | iterator i |
| **Return** | hash\_next()에 의해 가장 최근에 반환된 값 |
| **Function** | titeraiton에서의 최근 element를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | 크기를 알아낼 hash |
| **Return** | hash의 elements 수 |
| **Function** | h에 있는 elements의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h) |
| **Parameter** | 비어있는지 확인 할 hash |
| **Return** | hash안에 element가 없다면 true, 있다면 false |
| **Function** | hash가 비어있는지 여부를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size) |
| **Parameter** | buf, 반환할 size |
| **Return** | size bytes만큼의 hash |
| **Function** | buf에서 size만큼의 hash를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_) |
| **Parameter** | hash를 반환할 string s |
| **Return** | s의 hash |
| **Function** | s의 hash를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i) |
| **Parameter** | hash를 반환할 int i |
| **Return** | i의 hash |
| **Function** | i의 hash를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | #define hash\_entry(hash\_elem, struct, member) |
| **Parameter** | hash element, hash element가 들어있는 struct, hash element의 멤버명 |
| **Return** | struct 주소 |
| **Function** | hash element가 들어있는 structure의 포인터를 반환한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | 생성할 bitmap의 크기 |
| **Return** | 생성된 bitmap, 생성하지 못하면 NULL |
| **Function** | bit\_cnt 크기를 갖는 bitmap을 생성하여 0으로 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | 제거할 bitmap |
| **Return** | 없음 |
| **Function** | bitmap b를 제거한다 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | 크기를 알아낼 bitmap |
| **Return** | b의 bit수 |
| **Function** | b의 bit수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value) |
| **Parameter** | 값을 설정할 bitmap, 인덱스, 설정할 값 |
| **Return** | 없음 |
| **Function** | b의 idx위치의 값을 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | bit를 1로 마크할 bitmap, 인덱스 |
| **Return** | 없음 |
| **Function** | v의 idx위치의 값을 1로 마크한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 0으로 설정할 bitmap, 인덱스 |
| **Return** | 없음 |
| **Function** | b의 idx위치의 값을 0으로 리셋한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx) |
| **Parameter** | 값을 뒤집을 bitmap, 인덱스 |
| **Return** | 없음 |
| **Function** | b의 idx위치의 값을 현재와 반대로 뒤집는다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test(const struct bitmap \*b, size\_t idx) |
| **Parameter** | 값을 확인할 bitmap, 인덱스 |
| **Return** | b의 idx위치의 값 |
| **Function** | b의 idx위치의 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | 모든 값을 설정할 bitmap, 설정하려는 Boolean(true or false) |
| **Return** | 없음 |
| **Function** | b의 모든 값을 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 값을 설정할 bitmap, 인덱스, 설정하려는 Boolean(true or false) |
| **Return** | 없음 |
| **Function** | b의 start부터 시작하여 cnt만큼의 값을 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 특정 value를 가진 bit의 수를 알아낼 bitmap, 시작 인덱스, 탐색 개수, 확인할 Boolean(true or false) |
| **Return** | start부터 start+cnt까지에서 value에 해당하는 bit의 개수 |
| **Function** | b에서 start부터 start+cnt까지의 범위에서 value에 해당하는 값을 가지는 bit의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 탐색할 bitmap, 시작위치, 탐색 개수, 확인할 Boolean(true or false) |
| **Return** | b의 start~start+cnt에 value에 해당하는 bit가 있다면 true 아니면 false |
| **Function** | b의 start~start+cnt에 value에 해당하는 bit가 포함되어 있는지 여부를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 탐색할 bitmap, 시작위치, 탐색 개수 |
| **Return** | b의 start~start+cnt에 1이 하나라도 있으면 true, 아니면 false |
| **Function** | b의 start~start+cnt에 1이 하나라도 있는지 여부를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 탐색할 bitmap, 시작위치, 탐색 개수 |
| **Return** | b의 start~start+cnt에 1이 없으면 true, 하나라도 있으면 false |
| **Function** | b의 start~start+cnt에 1이 존재하지 않는가를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | 탐색할 bitmap, 시작위치, 탐색 개수 |
| **Return** | b의 start~start+cnt에 해당하는 bit가 전부 1이면 true, 아니면 false |
| **Function** | b의 start~start+cnt범위의 bit가 전부 1인지 여부를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 탐색할 bitmap, 시작위치, 탐색 개수, 확인할 Boolean |
| **Return** | b에서 cnt만큼 value가 연속한 범위의 첫번째 index |
| **Function** | value가 cnt만큼 처음으로 연속한 범위의 첫번째 index를 찾아내어 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | 탐색할 bitmap, 시작위치, 탐색 개수, 확인할 Boolean |
| **Return** | b에서 cnt만큼 value가 연속한 범위의 첫번째 index |
| **Function** | value가 cnt만큼 처음으로 연속한 범위의 첫번째 index를 찾아내어 반환하고 해당하는 bit들을 모두 반대로 뒤집는다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | 출력할 bitmap |
| **Return** | 없음 |
| **Function** | bitmap을 출력한다. |