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

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 교수님 |
| 학번 : | 20181593 |
| 이름 : | 계인혜 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | bool listCmp(const struct list\_elem \*a, const struct list\_elem \*b, void \* aux) |
| **Parameter** | List\_elem 포인터, List\_elem 포인터, 보조 변수 |
| **Return** | a의 데이터가 b의 데이터 보다 작으면 true를 아니면 false를 리턴한다. |
| **Function** | List\_elem의 데이터의 크기 비교를 수행한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hashFunc(const struct hash\_elem \*a, void \*aux) |
| **Parameter** | Hash\_elem 포인터, 보조 변수 |
| **Return** | A가 가리키는 데이터의 해싱 키를 리턴한다. |
| **Function** | A가 가리키는 데이터의 해싱 키를 계산하는 함수를 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hashCmp(const struct hash\_elem \*a, const struct hash\_elem \*b, void \* aux) |
| **Parameter** | Hash\_elem 포인터, Hash\_elem 포인터, 보조 변수 |
| **Return** | a의 데이터가 b의 데이터 보다 작으면 true를 아니면 false를 리턴한다. |
| **Function** | hash\_elem의 데이터의 크기 비교를 수행한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem \*a, void\* aux) |
| **Parameter** | Hash\_elem 포인터, 보조 변수 |
| **Return** | None |
| **Function** | A의 데이터의 값을 제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem \*a, void\* aux) |
| **Parameter** | Hash\_elem 포인터, 보조 변수 |
| **Return** | None |
| **Function** | A의 데이터의 값을 세제곱한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | None |
| **Function** | List를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List에 연결된 List\_elem중 가장 앞 부분의 포인터를 리턴한다. |
| **Function** | List에 연결된 List\_elem중 가장 앞 부분을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*); |
| **Parameter** | List\_elem 포인터 |
| **Return** | List\_elem의 다음 원소의 주소 값을 리턴한다. |
| **Function** | List\_elem의 다음 원소를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | 마지막 원소의 포인터를 리턴한다.(tail의 포인터) |
| **Function** | List의 마지막 list\_elem(tail)을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*); |
| **Parameter** | List\_elem 포인터 |
| **Return** | List\_elem의 이전 원소의 포인터를 리턴한다. |
| **Function** | 전달된 list\_elem의 직전에 있는 원소를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List의 head의 포인터 값을 리턴한다. |
| **Function** | List의 head를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List의 마지막 원소의 포인터 값을 리턴한다. |
| **Function** | List의 tail을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*, struct list\_elem \*); |
| **Parameter** | 기준 원소의 포인터, 새로운 원소의 포인터 |
| **Return** | None |
| **Function** | 기준 원소의 포인터 앞에 새로운 원소를 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | Splice한 list\_elem을 저장하는 포인터, splice한 list\_elem의 첫 원소, splice한 list\_elem의 끝 원소 |
| **Return** | None |
| **Function** | First 부터 last까지를 before로 이동시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*); |
| **Parameter** | 리스트 포인터, List\_elem 포인터 |
| **Return** | None |
| **Function** | List의 맨 앞에 list\_elem을 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*); |
| **Parameter** | 리스트 포인터, List\_elem 포인터 |
| **Return** | None |
| **Function** | List의 맨 뒤에 list\_elem을 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*); |
| **Parameter** | List\_elem 포인터 |
| **Return** | 삭제한 원소의 다음 원소의 포인터 값(list\_elem->next)을 리턴한다. |
| **Function** | List에서 특정 list\_elem을 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List에서 삭제된 맨 앞의 list\_elem의 포인터 값을 리턴한다. |
| **Function** | List의 맨 앞에 있는 list\_elem을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List에서 삭제된 맨 뒤의 list\_elem의 포인터 값을 리턴한다. |
| **Function** | List의 맨 뒤에 있는 list\_elem을 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List의 맨 앞 list\_elem을 가리키는 포인터를 리턴한다. |
| **Function** | List의 맨 앞 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List의 맨 뒤 list\_elem을 가리키는 포인터를 리턴한다. |
| **Function** | List의 맨 뒤 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List의 크기 |
| **Function** | List의 크기를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | List가 비어있으면 true, 아니면 false를 리턴한다. |
| **Function** | List가 비어있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | 리스트 포인터 |
| **Return** | None |
| **Function** | 리스트의 순서를 거꾸로 바꾸어 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 리스트 포인터, list compare 함수, 보조 변수 |
| **Return** | None |
| **Function** | 리스트를 list\_less\_func에 따라 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 리스트 포인터, list\_elem 포인터, list compare 함수, 보조 변수 |
| **Return** | None |
| **Function** | List compare함수에 따라 list\_elem을 정렬하여 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux); |
| **Parameter** | 리스트 포인터, 리스트 포인터, list compare 함수, 보조 변수 |
| **Return** | None |
| **Function** | List\_less\_func에 의해 List에 중복되지 않은 원소들만 남긴다. duplicate에는 중복되어 사라진 원소들이 저장된다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 리스트 포인터, list compare 함수, 보조 변수 |
| **Return** | 최대 값을 갖는 list\_elem의 포인터를 리턴한다. |
| **Function** | List에서 list\_less\_func를 사용해 최대 값을 갖는 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 리스트 포인터, list compare 함수, 보조 변수 |
| **Return** | 최소 값을 갖는 list\_elem의 포인터를 리턴한다. |
| **Function** | List에서 list\_less\_func를 사용해 최소 값을 갖는 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | 바꾸려는 리스트 원소 두 개 |
| **Return** | None |
| **Function** | 리스트의 두 원소를 서로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | 무작위로 섞을 리스트 |
| **Return** | None |
| **Function** | List의 element를 무작위로 섞는다. |

|  |  |
| --- | --- |
| **Prototype** | list\_entry(LIST\_ELEM, STRUCT, MEMBER) |
| **Parameter** | List\_elem, struct, member |
| **Return** | List\_elem을 포함하는 List\_item의 주소를 리턴한다. |
| **Function** | List\_elem을 포함하는 List\_item의 주소를 구한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | 해시 함수에 대입할 정수 |
| **Return** | I에 대응하는 해시 값 |
| **Function** | I에 대응하는 해시 값을 구한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \* ); |
| **Parameter** | 해시 포인터, 해시 함수, 비교 함수, 보조 변수 |
| **Return** | 초기화에 성공하면 true, 아니면 false를 리턴한다. |
| **Function** | 해시를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 해시 포인터, 해시 액션 함수 |
| **Return** | None |
| **Function** | 해시의 모든 버킷 값을 clear한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 해시 포인터, 해시 액션 함수 |
| **Return** | None |
| **Function** | 해시 버킷의 메모리를 해제하고 해시를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 해시 포인터, 해시 액션 함수 |
| **Return** | None |
| **Function** | 해시의 모든 값에 대해 해시 액션 함수를 취한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*, struct hash \*); |
| **Parameter** | 해시 iterator, 해시 포인터 |
| **Return** | None |
| **Function** | Iterator가 hash의 첫 버킷을 가리키게 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*); |
| **Parameter** | 해시 iterator |
| **Return** | Iterator가 가리키는 다음 hash\_elem |
| **Function** | Iterator가 가리키던 것의 다음 hash\_elem를 가리키게 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 해시 포인터, hash\_elem 포인터 |
| **Return** | Hash에 hash\_elem이 이미 있다면 기존의 것을, 없으면 null을 리턴한다. |
| **Function** | Hash에 새로운 hash\_elem을 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 해시 포인터, hash\_elem 포인터 |
| **Return** | Hash에 hash\_elem이 있으면 hash\_elem을 없으면 NULL을 리턴한다. |
| **Function** | Hash에 hash\_elem을 replace한다. 기존에 있던 것은 빼고 다시 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 해시 포인터, hash\_elem 포인터 |
| **Return** | Hash에 hash\_elem이 있으면 hash\_elem을 없으면 NULL을 리턴한다. |
| **Function** | Hash에 특정 hash\_elem이 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 해시 포인터, hash\_elem 포인터 |
| **Return** | Hash에 hash\_elem이 있으면 지운 hash\_elem을 없으면 NULL을 리턴한다. |
| **Function** | Hash에 있는 특정 hash\_elem을 지운다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*); |
| **Parameter** | 해시 포인터 |
| **Return** | 해시의 크기 |
| **Function** | 해시의 크기를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*); |
| **Parameter** | 해시 포인터 |
| **Return** | 해시가 비어있으면 true, 아니면 false를 리턴한다. |
| **Function** | 해시가 비어있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int); |
| **Parameter** | 정수 |
| **Return** | 해싱한 결과 값을 리턴한다. |
| **Function** | Int를 해시 함수를 이용해 해싱한다. |

|  |  |
| --- | --- |
| **Prototype** | hash\_entry(HASH\_ELEM, STRUCT, MEMBER) |
| **Parameter** | Hash\_elem, struct, member |
| **Return** | Hash\_elem을 포함하는 Hash\_item의 주소를 리턴한다. |
| **Function** | Hash\_elem을 포함하는 Hash\_item의 주소를 구한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | Size 변경을 하고자 하는 비트맵과 확장 시킬 size |
| **Return** | 성공하면 확장된 비트맵의 주소 값을, 실패하면 NULL값을 리턴한다. |
| **Function** | Expand the given BITMAP to the SIZE. (backward expansion) |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | 생성하고자 하는 비트맵의 사이즈 |
| **Return** | 생성된 비트맵의 포인터 |
| **Function** | Bit\_cnt 크기의 비트맵을 생성하고 포인터를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*); |
| **Parameter** | 삭제하고자 하는 비트맵의 포인터 |
| **Return** | None |
| **Function** | 비트맵을 메모리에서 free한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*); |
| **Parameter** | 사이즈를 알고자 하는 비트맵의 const 포인터 |
| **Return** | 비트맵의 크기 |
| **Function** | 비트맵의 크기를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool); |
| **Parameter** | 비트맵 포인터와 set하고자 하는 index, bool값 |
| **Return** | None |
| **Function** | Bitmap의 idx에 해당하는 값을 bool로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터와 mark 하고자 하는 index |
| **Return** | None |
| **Function** | Bitmap의 idx에 해당하는 값을 true로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터와 reset 하고자 하는 index |
| **Return** | None |
| **Function** | Bitmap의 idx에 해당하는 값을 false로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터와 flip 하고자 하는 index |
| **Return** | None |
| **Function** | Bitmap의 idx에 해당하는 값을 반전시켜 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터와 test 하고자 하는 index |
| **Return** | Bitmap의 idx에 해당하는 값이 1이면 true를 0이면 false를 반환한다. |
| **Function** | Bitmap의 idx에 해당하는 값을 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool); |
| **Parameter** | 비트맵 포인터와 set 하고자 하는 값 |
| **Return** | None |
| **Function** | Bitmap을 모두 bool로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool ); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, set하고자 하는 개수, bool값 |
| **Return** | None |
| **Function** | Bitmap의 start부터 start+cnt-1까지를 bool값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기, bool값 |
| **Return** | Bitmap의 start부터 start+cnt-1까지 bool값에 해당하는 비트 수를 리턴한다. |
| **Function** | Bitmap의 start부터 start+cnt-1까지 bool에 해당하는 비트 수를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기, bool값 |
| **Return** | Bitmap의 start부터 start+cnt-1까지 bool에 해당하는 값이 있으면 true를 없으면 false를 리턴한다. |
| **Function** | Bitmap의 특정 구간에서 bool값의 존재 여부를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기 |
| **Return** | Bitmap의 start부터 start+cnt-1까지 1이 하나라도 있으면 true를 아니면 false를 리턴한다. |
| **Function** | Bitmap의 특정 구간에서 1의 존재 여부를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기 |
| **Return** | Bitmap의 start부터 start+cnt-1까지 모두 0이면 true를 아니면 false를 리턴한다. |
| **Function** | Bitmap의 특정 구간에서 1의 존재 여부를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기 |
| **Return** | Bitmap의 start부터 start+cnt-1까지 모두 1이면 true를 아니면 false를 리턴한다. |
| **Function** | Bitmap의 특정 구간이 모두 1인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기, bool |
| **Return** | Bitmap의 start이후 연속으로 cnt 개의 bool값을 갖는 부분의 인덱스 |
| **Function** | Bitmap내에 연속으로 cnt개의 bool값을 갖는 부분을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터와 시작 인덱스, 시작 인덱스로부터 확인하고자 하는 크기, bool |
| **Return** | Bitmap의 start이후 연속으로 cnt 개의 bool값을 갖는 부분의 인덱스 |
| **Function** | Bitmap내에 연속으로 cnt개의 bool값을 갖는 부분을 찾고, 있으면 반전시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*); |
| **Parameter** | 비트맵 포인터 |
| **Return** | None |
| **Function** | Bitmap의 값을 hex\_dump를 사용하여 16진수로 보여준다. |