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

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 |
| 학번 : | 20151378 |
| 이름 : | 진상우 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | Void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | Swap하려는 2개의 elements |
| **Return** | None |
| **Function** | Parameter로 넘어온 2개의 element를 swap한다 |

|  |  |
| --- | --- |
| **Prototype** | Void list\_shuffle(struct list \*list) |
| **Parameter** | Shuffle 하려는 list |
| **Return** | None |
| **Function** | Parameter로 넘어온 list의 element를 무작위로 섞는다. |

|  |  |
| --- | --- |
| **Prototype** | Unsigned hash\_int\_2(int i) |
| **Parameter** | Hash 값을 구하려는 integer |
| **Return** | Integer i의 hash 값 |
| **Function** | 자유롭게 만들고 문서에 기술 |

|  |  |
| --- | --- |
| **Prototype** | Struct bitmap \*bitmap\_expand (struct bitmap \*bitmap, int size) |
| **Parameter** | 확장하고 싶은 bitmap과 size |
| **Return** | 성공 시 확장된 bitmap, 실패 시 NULL |
| **Function** | 해당 bitmap의 길이를 size만큼 늘린다(뒤로 확장) |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | Void list\_init(struct list \*) |
| **Parameter** | List |
| **Return** | None |
| **Function** | List를 초기화 하여 다시 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*); |
| **Parameter** | List |
| **Return** | List의 처음 부분 |
| **Function** | List의 head.next 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*); |
| **Parameter** | List\_elem |
| **Return** | List\_elem의 다음 노드 |
| **Function** | List\_elem의 현재 노드의 바로 다음 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*); |
| **Parameter** | List |
| **Return** | List의 tail |
| **Function** | List의 마지막 노드 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*); |
| **Parameter** | List |
| **Return** | Tail의 prev |
| **Function** | 마지막 노드의 prev를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*); |
| **Parameter** | List\_elem |
| **Return** | List\_elem |
| **Function** | List\_elem의 이전 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*) |
| **Parameter** | List |
| **Return** | List의 head |
| **Function** | List의 head를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*) |
| **Parameter** | List |
| **Return** | List의 head |
| **Function** | List의 head를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*) |
| **Parameter** | List |
| **Return** | List의 tail |
| **Function** | List의 tail을 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*a, struct list\_elem \*b); |
| **Parameter** | List\_elem |
| **Return** | None |
| **Function** | List의 노드 a앞에 노드 b를 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | List\_elem |
| **Return** | None |
| **Function** | List안의 first부터 last까지 노드를 잘라 before 노드 앞에 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*); |
| **Parameter** | List,list\_elem |
| **Return** | none |
| **Function** | list와 list\_elem을 parameter로 받아서, list의 맨 앞에 list\_elem을 연결 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*) |
| **Parameter** | List,list\_elem |
| **Return** | none |
| **Function** | list와 list\_elem을 parameter로 받아서, list의 맨 뒤에 list\_elem를 연결 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*) |
| **Parameter** | List\_elem |
| **Return** | 다음 list\_elem |
| **Function** | 해당 list\_elem을 list에서 제거하고, 다음 list\_elem을 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*); |
| **Parameter** | list |
| **Return** | List의 가장 앞 노드 |
| **Function** | list의 가장 앞 노드를 제거하고, 그 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*); |
| **Parameter** | list |
| **Return** | List의 가장 뒷 노드 |
| **Function** | list의 가장 뒷 노드를 제거하고, 그 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*); |
| **Parameter** | list |
| **Return** | List의 처음 노드 |
| **Function** | list의 처음 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*) |
| **Parameter** | list |
| **Return** | List의 가장 마지막 노드 |
| **Function** | list의 가장 마지막 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*); |
| **Parameter** | list |
| **Return** | List 노드의 개수 |
| **Function** | List 노드의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*); |
| **Parameter** | list |
| **Return** | True/false |
| **Function** | list가 비어있으면 true, 비어있지 않으면 false를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | list |
| **Return** | None |
| **Function** | List의 순서를 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | typedef bool list\_less\_func (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux); |
| **Parameter** | List\_elem |
| **Return** | True/false |
| **Function** | a<b이면 true를, a=b이거나 a>b이면 false를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | list |
| **Return** | none |
| **Function** | List를 오름차순으로 정렬 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux); |
| **Parameter** | List, list\_elem |
| **Return** | none |
| **Function** | list에 list\_elem을 추가할 때 오름차순으로 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux); |
| **Parameter** | list |
| **Return** | none |
| **Function** | 그 list안에 중복된 data가 있으면, 그 값을 duplicate list로 넣어주고, list에서는 data당 노드 1개로 줄인다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | list |
| **Return** | 가장 큰 값을 가지는 노드 |
| **Function** | 가장 큰 값을 가지는 노드를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | list |
| **Return** | 가장 작은 값을 가지는 노드 |
| **Function** | 가장 작은 값을 가지는 노드를 반환 |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | Bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux); |
| **Parameter** | Hash |
| **Return** | none |
| **Function** | Hash 초기화 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*, hash\_action\_func \*); |
| **Parameter** | hash |
| **Return** | none |
| **Function** | Clear시켜준다 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*, hash\_action\_func \*); |
| **Parameter** | hash |
| **Return** | none |
| **Function** | Hash의 bucket을 free |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*); |
| **Parameter** | Hash, hash\_elem |
| **Return** | none |
| **Function** | hash에 hash\_elem을 삽입 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*); |
| **Parameter** | Hash, hash\_elem |
| **Return** | none |
| **Function** | hash에 hash\_elem을 삽입하고 같은 element라면 새로운 것으로 대체 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*); |
| **Parameter** | Hash,hash\_elem |
| **Return** | Hash의 위치값 |
| **Function** |  |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*); |
| **Parameter** | Hash,hash\_elem |
| **Return** | none |
| **Function** | hash\_elem이 hash에 어느 곳에 있는지 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*, hash\_action\_func \*); |
| **Parameter** | hash |
| **Return** | none |
| **Function** | hash의 모든 데이터에 대해서 hash\_action\_func을 수행 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*); |
| **Parameter** | hash |
| **Return** | Hash의 크기 |
| **Function** | Hash의 크기 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*); |
| **Parameter** | hash |
| **Return** | True/false |
| **Function** | hash가 비어있으면 true, 비어있지 않다면 false를 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int); |
| **Parameter** | int |
| **Return** | Int의 hash값 |
| **Function** | int의 hash값을 반환 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | Bit\_cnt |
| **Return** | Bitmap |
| **Function** | Bit\_cnt의 크기만큼 bitmap생성 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*); |
| **Parameter** | bitmap |
| **Return** | none |
| **Function** | 받은 bitmap을 삭제 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*); |
| **Parameter** | Bitmap |
| **Return** | Bitmap의 size |
| **Function** | 받은 bitmap의 size를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool); |
| **Parameter** | Bitmap, idx |
| **Return** | none |
| **Function** | Bitmap의 idx번째 bit를 value로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap, idx |
| **Return** | true |
| **Function** | Bitmap의 idx번째 bit를 true로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap, idx |
| **Return** | false |
| **Function** | Bitmap의 idx번째 bit를 false로 설정 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap, idx |
| **Return** | True/false |
| **Function** | bitmap의 idx번째 bit를 true이면 false로, false면 true로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| **Parameter** | Bitmap, idx |
| **Return** | Idx위치의 value |
| **Function** | idx의 위치의 value를 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool); |
| **Parameter** | bitmap |
| **Return** | none |
| **Function** | , bitmap의 전체 bit를 value로 값을 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, sizt\_t cnt, bool); |
| **Parameter** | Bitmap,start,cnt |
| **Return** | none |
| **Function** | start부터 cnt까지 bit를 value로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | Bitmap,start,cnt |
| **Return** | 특정 bool의 개수 |
| **Function** | start부터 cnt까지 특정 bool의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt,bool); |
| **Parameter** | Bitmap,start,cnt |
| **Return** | True/false |
| **Function** | start부터 cnt까지 특정 bool을 가진 bit가 하나라도 있으면 true를, 아니면 false를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | Bitmap,start,cnt |
| **Return** | True/false |
| **Function** | start부터 cnt까지 true가 하나라도 있으면 true를, 하나도 없으면 false를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | Bitmap,start,cnt |
| **Return** | true |
| **Function** | start부터 cnt까지 모두 bit가 false이면 true를 반환 |

|  |  |
| --- | --- |
| **Prototype** | **bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt);** |
| **Parameter** | Bitmap,start,cnt |
| **Return** | true |
| **Function** | start부터 cnt까지 모두 bit가 true이면 true를 반환 |

|  |  |
| --- | --- |
| **Prototype** | **size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool);** |
| **Parameter** | Bitmap,start,cnt |
| **Return** | true |
| **Function** | start부터 cnt만큼 연속으로 bit가 bool값을 가지면 true를 반환 |

|  |  |
| --- | --- |
| **Prototype** | **size\_t bitmap\_scan\_and\_flip (struct bitmap \*,size\_t start, size\_t cnt, bool);** |
| **Parameter** | Bitmap,start,cnt |
| **Return** | False/true |
| **Function** | bitmap\_scan과 동일한 작업을 한 후, 각각의 bit의 value를 true이면 false로, false이면 true로 바꿈 |

|  |  |
| --- | --- |
| **Prototype** | **void bitmap\_dump (const struct bitmap \*);** |
| **Parameter** | bitmap |
| **Return** | 16진수의 bitmap |
| **Function** | Bitmap을 16진수로 출력 |