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

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 |
| 학번 : | 20181652 |
| 이름 : | 양희원 |
|  |  |

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | int  read\_command**(**char**\*\*** command\_token**,** int token\_cnt**)** |
| **Parameter** | char**\*\*** command\_token : 입력 받은 명령어를 ‘ ‘를 기준으로 나눈 문자열 배열  int token\_cnt : 나눈 문자열의 수 |
| **Return** | command\_token[0]이 “quit”인 경우 0, 아니면 1 |
| **Function** | 입력 받은 명령어에 따라 알맞은 실행을 한다. |

|  |  |
| --- | --- |
| **Prototype** | char**\*\***  tokenize\_command**(**char**\*** command**,** int**\*** token\_cnt**)** |
| **Parameter** | char**\*** command : 입력 받은 명령어 문자열  int**\*** token\_cnt : 나눌 문자열의 수 |
| **Return** | ‘ ‘ 를 기준으로 나눈 문자열 배열 |
| **Function** | 입력 받은 명령어를 ‘ ‘를 기준으로 나누어 문자열 배열을 반환한다. 문자열 배열의 수를 token\_cnt에 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  list\_less**(**const struct list\_elem**\*** a**,** const struct list\_elem**\*** b**,** void**\*** aux**)** |
| **Parameter** | const struct list\_elem**\*** a**,** const struct list\_elem**\*** b : 서로 data 값을 비교하려는 element  void**\*** aux : 보조 값 |
| **Return** | element의 data 값이 a< b 인 경우 true, 아니면 false |
| **Function** | parameter로 받은 list element의 data값을 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem**\***  find\_elem**(**int list\_idx**,** int item\_idx**)** |
| **Parameter** | int list\_idx : 찾으려는 list의 index  int item\_idx : 찾으려는 list item의 index |
| **Return** | list\_idx의 list에서 item\_idx의 item에 해당하는 element 주소 |
| **Function** | list\_idx의 list에서 item\_idx의 item에 해당하는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned  hash\_hash**(**const struct hash\_elem**\*** a**,** void**\*** aux**)** |
| **Parameter** | const struct hash\_elem**\*** a : hash key를 찾으려는 element  void**\*** aux : 보조 값 |
| **Return** | hash key |
| **Function** | parameter로 받은 element를 찾아 hash key를 계산하여 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  hash\_less**(**const struct hash\_elem**\*** a**,** const struct hash\_elem**\*** b**,** void**\*** aux |
| **Parameter** | const struct hash\_elem**\*** a**,** const struct hash\_elem**\*** b : 서로 data 값을 비교하려는 element  void**\*** aux : 보조 값 |
| **Return** | element의 data 값이 a< b 인 경우 true, 아니면 false |
| **Function** | parameter로 받은 hash element의 data값을 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void  destructor**(**struct hash\_elem**\*** a**,** void**\*** aux**)** |
| **Parameter** | struct hash\_elem**\*** a : 삭제하려는 element  void**\*** aux : 보조 값 |
| **Return** | None |
| **Function** | paramter로 받은 element를 hash에서 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_print**(**struct hash\_elem**\*** a**,** void**\*** aux**)** |
| **Parameter** | struct hash\_elem**\*** a : 출력하려는 hash  void**\*** aux : 보조 값 |
| **Return** | None |
| **Function** | element의 data 값을 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_square**(**struct hash\_elem**\*** a**,** void**\*** aux**)** |
| **Parameter** | struct hash\_elem**\*** a : data 값을 제곱하려는 element  void**\*** aux : 보조 값 |
| **Return** | None |
| **Function** | element의 data 값을 제곱하여 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_triple**(**struct hash\_elem**\*** a**,** void**\*** aux) |
| **Parameter** | struct hash\_elem**\*** a : data 값을 세제곱하려는 element  void**\*** aux : 보조 값 |
| **Return** | None |
| **Function** | element의 data 값을 세제곱하여 저장한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | static inline bool  is\_head **(**struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***elem : head인지 확인하려는 list element |
| **Return** | head인 경우 true, 아니면 false |
| **Function** | element가 list의 head인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool  is\_interior **(**struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***elem : interior인지 확인하려는 list element |
| **Return** | interior인 경우 true, 아니면 false |
| **Function** | element가 list의 interior(내부 요소)인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool  is\_tail **(**struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***elem : tail인지 확인하려는 list element |
| **Return** | tail인 경우 true, 아니면 false |
| **Function** | element가 list의 tail인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_init **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 초기화하려는 list |
| **Return** | None |
| **Function** | list를 empty list로 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_begin **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 시작 element 주소 |
| **Function** | list의 시작 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_next **(**struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***elem : 찾으려는 element의 이전 element 주소 |
| **Return** | parameter로 받은 element의 다음 element 주소 |
| **Function** | parameter로 받은 element의 다음 element 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_end **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 tail에 해당하는 element의 주소 |
| **Function** | list의 마지막 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_rbegin **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 역방향 시작 element의 주소 |
| **Function** | list를 역순으로 하기 위해 역방향 시작 element의 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_prev **(**struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***elem : 찾으려는 element의 다음 element 주소 |
| **Return** | parameter로 받은 element의 이전 element 주소 |
| **Function** | parameter로 받은 element의 이전 element 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_rend **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 역방향 끝 element의 주소 |
| **Function** | list를 역순으로 하기 위해 역방향 끝 element의 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_head **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 head에 해당하는 element의 주소 |
| **Function** | list의 head에 해당하는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_tail **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 tail에 해당하는 element의 주소 |
| **Function** | list의 tail에 해당하는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_insert **(**struct list\_elem **\***before**,** struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***before : 삽입하려는 element의 위치 기준 element  struct list\_elem **\***elem : list에 삽입하려는 새 element |
| **Return** | None |
| **Function** | parameter로 받은 before의 이전(prev)에 elem을 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_splice **(**struct list\_elem **\***before**,** struct list\_elem **\***first**,** struct list\_elem **\***last**)** |
| **Parameter** | struct list\_elem **\***before : splice하려는 element들의 위치 기준 element  struct list\_elem **\***first : splice하려는 element들의 시작 element  struct list\_elem **\***last : splice하려는 element들의 끝 element |
| **Return** | None |
| **Function** | first부터 last까지의 element들을 before의 이전에 끼워 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_push\_front **(**struct list **\***list**,** struct list\_elem **\***elem**)** |
| **Parameter** | struct list **\***list : 삽입하려는 list  struct list\_elem **\***elem : 삽입하려는 element |
| **Return** | None |
| **Function** | list의 맨 앞에 element를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_push\_back **(**struct list **\***list**,** struct list\_elem **\***elem**)** |
| **Parameter** | struct list **\***list : 삽입하려는 list  struct list\_elem **\***elem : 삽입하려는 element |
| **Return** | None |
| **Function** | list의 맨 뒤에 element를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_remove **(**struct list\_elem **\***elem**)** |
| **Parameter** | struct list\_elem **\***elem : 삭제하려는 element |
| **Return** | parameter로 받은 element->next 값 |
| **Function** | list에서 element를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_pop\_front **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : pop하려는 element의 해당 list |
| **Return** | pop한 element의 주소 |
| **Function** | list의 맨 앞에 있는 element를 list에서 삭제한 뒤 해당 element의 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_pop\_back **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : pop하려는 element의 해당 list |
| **Return** | pop한 element의 주소 |
| **Function** | list의 맨 뒤에 있는 element를 list에서 삭제한 뒤 해당 element의 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_front **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 맨 앞에 있는 element의 주소 |
| **Function** | list의 맨 앞에 있는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_back **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list |
| **Return** | list의 맨 뒤에 있는 element의 주소 |
| **Function** | list의 맨 뒤에 있는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  list\_size **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : 크기를 구하려는 list |
| **Return** | list의 크기 (element의 수) |
| **Function** | list의 크기를 계산하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  list\_empty **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : empty한지 확인하려는 list |
| **Return** | list가 비어 있다면 true, 아니면 false |
| **Function** | list가 빈 list인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static void  swap **(**struct list\_elem **\*\***a**,** struct list\_elem **\*\***b**)** |
| **Parameter** | struct list\_elem **\*\***a**,** struct list\_elem **\*\***b : swap하려는 두 개의 list\_element를 가리키는 포인터들 |
| **Return** | None |
| **Function** | a와 b가 가리키는 struct list\_elem **\***를 swap한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_reverse **(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : reverse 하려는 list |
| **Return** | None |
| **Function** | list의 순서를 reverse한다. |

|  |  |
| --- | --- |
| **Prototype** | static bool  is\_sorted **(**struct list\_elem **\***a**,** struct list\_elem **\***b**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list\_elem **\***a : 정렬을 확인하려는 범위의 시작 element  struct list\_elem **\***b : 정렬을 확인하려는 범위의 끝 element  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | a부터 b까지 element들이 정렬이 되어 있다면 true, 아니면 false |
| **Function** | a부터 b까지 element들이 less 함수에 맞게 정렬이 되어 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem **\***  find\_end\_of\_run **(**struct list\_elem **\***a**,** struct list\_elem **\***b**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list\_elem **\***a : 확인하려는 범위의 시작 element  struct list\_elem **\***b : 확인하려는 범위의 끝 element  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | 순서가 맞지 않는 element의 주소 |
| **Function** | a부터 b까지 element 중에서 less 함수에 따라 이전 element와 순서가 맞지 않는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static void  inplace\_merge **(**struct list\_elem **\***a0**,** struct list\_elem **\***a1b0**,** struct list\_elem **\***b1**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list\_elem **\***a0**,** struct list\_elem **\***a1b0**,** struct list\_elem **\***b1 : merge하려는 범위의 시작과 끝 element들  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | None |
| **Function** | a0~a1b0 와 a1b0~ b1 를 merge하여 로 끝나는 결합 범위를 형성한다. 이때 입력 범위는 비어 있지 않아야 하고 less 함수에 따른 순서로 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_sort **(**struct list **\***list**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list **\***list : 정렬하려는 list  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | None |
| **Function** | less 함수에 따라 list를 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_insert\_ordered **(**struct list **\***list**,** struct list\_elem **\***elem**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list **\***list : 삽입하려는 list  struct list\_elem **\***elem : 삽입하려는 element  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | None |
| **Function** | parameter로 받은 element를 less 함수에 따라 정렬이 유지되도록 list에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_unique **(**struct list **\***list**,** struct list **\***duplicates**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list **\***list : 확인하려는 list  struct list **\***duplicates : 중복된 element를 저장하는 list  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | None |
| **Function** | list에 값이 중복되는 element가 있으면 하나를 삭제하고 duplicates list의 맨 뒤에 해당 element를 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_max **(**struct list **\***list**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | list에서 가장 큰 data를 가진 element의 주소 |
| **Function** | list에서 less 함수를 기준으로 가장 큰 data 값을 갖는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem **\***  list\_min **(**struct list **\***list**,** list\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct list **\***list : 찾으려는 element의 해당 list  list\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | list에서 가장 작은 data를 가진 element의 주소 |
| **Function** | list에서 less 함수를 기준으로 가장 작은 data 값을 갖는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_swap**(**struct list\_elem **\***a**,** struct list\_elem **\***b**)** |
| **Parameter** | struct list\_elem **\***a**,** struct list\_elem **\***b : 서로 swap하려는 두 element |
| **Return** | None |
| **Function** | parameter로 받은 두 element를 swap한다. |

|  |  |
| --- | --- |
| **Prototype** | void  list\_shuffle**(**struct list **\***list**)** |
| **Parameter** | struct list **\***list : shuffle하려는 list |
| **Return** | None |
| **Function** | list의 element를 무작위로 섞는다. |

|  |  |
| --- | --- |
| **Prototype** | list\_entry(LIST\_ELEM, STRUCT, MEMBER) |
| **Parameter** | LIST\_ELEM : 가리키는 element  STRUCT : 가리키고자 하는 구조  MEMBER : element의 멤버 |
| **Return** | parameter로 받은 LIST\_ELEM을 가진 STRUCT에 대한 주소 |
| **Function** | LIST\_ELEM이 내부에 내장되어 있는 STRUCT에 대한 포인터로 변환한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool  hash\_init **(**struct hash **\***h**,** hash\_hash\_func **\***hash**,** hash\_less\_func **\***less**,** void **\***aux**)** |
| **Parameter** | struct hash **\***h : 초기화하려는 hash  hash\_hash\_func **\***hash : hashing을 위한 함수  hash\_less\_func **\***less : element의 data를 비교하는 함수  void **\***aux : 보조 값 |
| **Return** | hash가 초기화가 되면 true, 아니면 false |
| **Function** | hash를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_clear **(**struct hash **\***h**,** hash\_action\_func **\***destructor**)** |
| **Parameter** | struct hash **\***h : clear하려는 hash  hash\_action\_func **\***destructor : hash element를 deallocate하는 함수 |
| **Return** | None |
| **Function** | hash의 모든 element에 destructor 함수를 적용해 clear한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_destroy **(**struct hash **\***h**,** hash\_action\_func **\***destructor**)** |
| **Parameter** | struct hash **\***h : destroy하려는 hash  hash\_action\_func **\***destructor : hash element를 deallocate하는 함수 |
| **Return** | None |
| **Function** | hash를 clear한 후 bucket을 free한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem **\***  hash\_insert **(**struct hash **\***h**,** struct hash\_elem **\***new**)** |
| **Parameter** | struct hash **\***h : 삽입하려는 hash  struct hash\_elem **\***new : 삽입하려는 element |
| **Return** | hash에 이미 같은 data 값을 갖는 element가 있으면 그 element의 주소, 아니면 NULL |
| **Function** | hash에 새 element를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem **\***  hash\_replace **(**struct hash **\***h**,** struct hash\_elem **\***new**)** |
| **Parameter** | struct hash **\***h : 삽입하려는 hash  struct hash\_elem **\***new : 삽입하려는 element |
| **Return** | hash에 이미 같은 data 값을 갖는 element가 있으면 그 element의 주소, 아니면 NULL |
| **Function** | hash에 새 element를 삽입한다. hash에 이미 같은 data 값을 갖는 element(old)가 있으면 이를 hash에서 삭제하고 새 element를 추가한 뒤 이전 element(old)의 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem **\***  hash\_find **(**struct hash **\***h**,** struct hash\_elem **\***e**)** |
| **Parameter** | struct hash **\***h : 찾으려는 hash  struct hash\_elem **\***e : 찾으려는 element |
| **Return** | parameter로 받은 element의 data 값과 같은 data 값을 갖는 element의 주소 |
| **Function** | parameter로 받은 element의 data 값과 같은 data 값을 갖는 element를 찾아 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem **\***  hash\_delete **(**struct hash **\***h**,** struct hash\_elem **\***e**)** |
| **Parameter** | struct hash **\***h : 삭제하려는 hash  struct hash\_elem **\***e : 삭제하려는 element |
| **Return** | 삭제하려는 element의 주소 |
| **Function** | parameter로 받은 element의 data 값과 같은 data 값을 갖는 element를 찾아 해당 element를 hash에서 제거하고 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_apply **(**struct hash **\***h**,** hash\_action\_func **\***action**)** |
| **Parameter** | struct hash **\***h : 적용하려는 hash  hash\_action\_func **\***action : 적용하려는 action 함수 |
| **Return** | None |
| **Function** | hashtable의 모든 element에 parameter로 받은 action 함수를 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void  hash\_first **(**struct hash\_iterator **\***i**,** struct hash **\***h**)** |
| **Parameter** | struct hash\_iterator **\***i : iterator  struct hash **\***h : 가리키려는 hash |
| **Return** | None |
| **Function** | parameter로 받은 iterator가 hash를 가리키게 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem **\***  hash\_next **(**struct hash\_iterator **\***i**)** |
| **Parameter** | struct hash\_iterator **\***i : iterator |
| **Return** | iterator가 next로 가리키고 있는 element 주소 |
| **Function** | parameter로 받은 iterator가 next로 가리키고 있는 element 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem **\***  hash\_cur **(**struct hash\_iterator **\***i**)** |
| **Parameter** | struct hash\_iterator **\***i : iterator |
| **Return** | iterator가 가리키고 있는 element 주소 |
| **Function** | parameter로 받은 iterator가 현재 가리키고 있는 element 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  hash\_size **(**struct hash **\***h**)** |
| **Parameter** | struct hash **\***h : 크기를 구하려는 hash |
| **Return** | hash의 크기 (element의 수) |
| **Function** | hash의 크기를 계산하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  hash\_empty **(**struct hash **\***h**)** |
| **Parameter** | struct hash **\***h : empty인지 확인하려는 hash |
| **Return** | hash가 비어 있다면 true, 아니면 false |
| **Function** | hash가 빈 hash인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned  hash\_bytes **(**const void **\***buf\_**,** size\_t size**)** |
| **Parameter** | const void **\***buf\_ : 버퍼  size\_t size : 크기 |
| **Return** | buf\_ 안에서 hash가 차지하는 byte size |
| **Function** | hash key 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned  hash\_string **(**const char **\***s\_**)** |
| **Parameter** | const char **\***s\_ : hash 값을 구하려는 문자열 |
| **Return** | 문자열의 hash 값 |
| **Function** | 문자열의 hash 값을 계산하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned  hash\_int **(**int i**)** |
| **Parameter** | int i : hash 값을 구하려는 정수 |
| **Return** | 정수의 hash 값 |
| **Function** | 정수의 hash 값을 계산하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list **\***  find\_bucket **(**struct hash **\***h**,** struct hash\_elem **\***e**)** |
| **Parameter** | struct hash **\***h : 찾으려는 bucket의 hash  struct hash\_elem **\***e : 찾으려는 bucket에 속한 element |
| **Return** | parameter로 받은 hash의 element가 속한 bucket |
| **Function** | parameter로 받은 hash의 element가 속한 bucket을 찾아 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem **\***  find\_elem **(**struct hash **\***h**,** struct list **\***bucket**,** struct hash\_elem **\***e**)** |
| **Parameter** | struct hash **\***h : 찾으려는 element의 해당 hash  struct list **\***bucket : 찾으려는 element의 해당 bucket  struct hash\_elem **\***e : 찾으려는 element |
| **Return** | parameter로 받은 hash의 bucket 안 element의 data 값과 같은 data 값을 갖는 element |
| **Function** | parameter로 받은 hash의 bucket 안 element의 data 값과 같은 data 값을갖는 element를 찾아 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t  turn\_off\_least\_1bit **(**size\_t x**)** |
| **Parameter** | size\_t x : 값을 바꿀 변수 |
| **Return** | 가장 낮은 bit가 turn off된 x |
| **Function** | x의 가장 낮은 bit 하나를 turn off 해서 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t  is\_power\_of\_2 **(**size\_t x**)** |
| **Parameter** | size\_t x : 확인하려는 변수 |
| **Return** | x가 2의 지수승이면 true, 아니면 false |
| **Function** | x가 2의 지수승인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static void  rehash **(**struct hash **\***h**)** |
| **Parameter** | struct hash **\***h : rehash하려는 hash |
| **Return** | None |
| **Function** | hash를 이상적으로 맞추기 위해 bucket의 수를 조정하여 다시 hash한다. |

|  |  |
| --- | --- |
| **Prototype** | static void  insert\_elem **(**struct hash **\***h**,** struct list **\***bucket**,** struct hash\_elem **\***e**)** |
| **Parameter** | struct hash **\***h : 삽입하려는 hash  struct list **\***bucket : 삽입하려는 bucket  struct hash\_elem **\***e : 삽입하려는 element |
| **Return** | None |
| **Function** | parameter로 받은 hash의 bucket 안에 element를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | static void  remove\_elem **(**struct hash **\***h**,** struct hash\_elem **\***e**)** |
| **Parameter** | struct hash **\***h : 삭제하려는 hash  struct hash\_elem **\***e : 삭제하려는 element |
| **Return** | None |
| **Function** | parameter로 받은 hash에서 element를 삭제한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned  hash\_int\_2**(**int x**)** |
| **Parameter** | int x : hash 값을 구하려는 정수 |
| **Return** | 정수의 hash 값 |
| **Function** | 정수의 hash 값을 계산하여 반환한다.  hashtable의 크기는 1000으로 설정하여 hash 값은 0~999에 해당한다.  x **=** **((**x**>>**16**)^**x**)\***0x1111**;**  x **=** **((**x**>>**16**)^**x**);**  x **=** x**%**1000**;**  **return** x**;** |

|  |  |
| --- | --- |
| **Prototype** | list\_elem\_to\_hash\_elem(LIST\_ELEM) |
| **Parameter** | LIST\_ELEM : 가리키는 element |
| **Return** | parameter로 받은 LIST\_ELEM을 기반으로 한 HASH\_ELEM |
| **Function** | LIST\_ELEM을 HASH\_ELEM으로 변경한다. |

|  |  |
| --- | --- |
| **Prototype** | hash\_entry(HASH\_ELEM, STRUCT, MEMBER) |
| **Parameter** | HASH\_ELEM : 가리키는 element  STRUCT : 가리키고자 하는 구조  MEMBER : element의 멤버 |
| **Return** | parameter로 받은 HASH\_ELEM을 가진 STRUCT에 대한 주소 |
| **Function** | HASH\_ELEM이 내부에 내장되어 있는 STRUCT에 대한 포인터로 변환한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t  elem\_idx **(**size\_t bit\_idx**)** |
| **Parameter** | size\_t bit\_idx : 찾으려는 bit index |
| **Return** | bit\_idx의 bit를 포함한 element의 index |
| **Function** | bit\_idx의 bit를 포함한 element의 index를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type  bit\_mask **(**size\_t bit\_idx**)** |
| **Parameter** | size\_t bit\_idx : 확인하려는 bit index |
| **Return** | elem\_type |
| **Function** | bit\_idx에 상응하는 bit가 turn on일 때 elem\_type을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t  elem\_cnt **(**size\_t bit\_cnt**)** |
| **Parameter** | size\_t bit\_cnt : bit count |
| **Return** | bit\_cnt bits를 위해 필요한 element의 수 |
| **Function** | bit\_cnt bits를 위해 필요한 element의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t  byte\_cnt **(**size\_t bit\_cnt**)** |
| **Parameter** | size\_t bit\_cnt : bit count |
| **Return** | bit\_cnt bits를 위해 필요한 bytes의 수 |
| **Function** | bit\_cnt bits를 위해 필요한 bytes의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type  last\_mask **(**const struct bitmap **\***b**)** |
| **Parameter** | const struct bitmap **\***b : 확인하려는 비트맵 |
| **Return** | b‘s bits의 마지막 element에 실제로 사용된 bit가 1로 설정되고 나머지는 0으로 설정된 bit mask |
| **Function** | b‘s bits의 마지막 element에 실제로 사용된 bit가 1로 설정되고 나머지는 0으로 설정된 bit mask를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap **\***  bitmap\_create **(**size\_t bit\_cnt**)** |
| **Parameter** | size\_t bit\_cnt : 생성하려는 bitmap의 크기 |
| **Return** | 생성한 bitmap의 주소 |
| **Function** | bit\_cnt 크기의 bitmap을 생성하고 그 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap **\***  bitmap\_create\_in\_buf **(**size\_t bit\_cnt**,** void **\***block**,** size\_t block\_size **)** |
| **Parameter** | size\_t bit\_cnt : 생성하려는 bitmap의 크기  void **\***block : 메모리 공간  size\_t block\_size : 미리 할당된 크기 |
| **Return** | 생성한 bitmap의 주소 |
| **Function** | block에 미리 할당된 block\_size 크기의 공간 안에 bit\_cnt 크기의 bitmap을 생성하고 그 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  bitmap\_buf\_size **(**size\_t bit\_cnt**)** |
| **Parameter** | size\_t bit\_cnt : bit count |
| **Return** | bit\_cnt bits와 bitmap을 수용하는 데에 필요한 bytes의 수 |
| **Function** | bit\_cnt bits와 bitmap을 수용하는 데에 필요한 bytes의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_destroy **(**struct bitmap **\***b**)** |
| **Parameter** | struct bitmap **\***b : free하려는 bitmap |
| **Return** | None |
| **Function** | bitmap을 free한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  bitmap\_size **(**const struct bitmap **\***b**)** |
| **Parameter** | const struct bitmap **\***b : 구하려는 bitmap |
| **Return** | bitmap의 bit의 수 |
| **Function** | bitmap의 bit의 수(bit\_cnt)를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_set **(**struct bitmap **\***b**,** size\_t idx**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 설정하려는 bitmap  size\_t idx : 설정하려는 bit의 index  bool value : 설정하려는 값 |
| **Return** | None |
| **Function** | bitmap의 idx의 bit에 value를 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_mark **(**struct bitmap **\***b**,** size\_t bit\_idx**)** |
| **Parameter** | struct bitmap **\***b : 설정하려는 bitmap  size\_t idx : 설정하려는 bit의 index |
| **Return** | None |
| **Function** | bitmap의 idx의 bit에 true(1)를 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_reset **(**struct bitmap **\***b**,** size\_t bit\_idx**)** |
| **Parameter** | struct bitmap **\***b : 설정하려는 bitmap  size\_t idx : 설정하려는 bit의 index |
| **Return** | None |
| **Function** | bitmap의 idx의 bit에 false(0)를 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_flip **(**struct bitmap **\***b**,** size\_t bit\_idx**)** |
| **Parameter** | struct bitmap **\***b : 설정하려는 bitmap  size\_t idx : 설정하려는 bit의 index |
| **Return** | None |
| **Function** | bitmap의 idx의 bit에 기존의 값과 반대인 값을 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_test **(**const struct bitmap **\***b**,** size\_t idx**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t idx : 확인하려는 bit의 index |
| **Return** | idx의 bit의 값이 true면 true, 아니면 false |
| **Function** | 비트맵의 idx의 bit 값을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_set\_all **(**struct bitmap **\***b**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 설정하려는 bitmap  bool value : 설정하려는 값 |
| **Return** | None |
| **Function** | bitmap의 모든 bit를 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_set\_multiple **(**struct bitmap **\***b**,** size\_t start**,** size\_t cnt**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 설정하려는 bitmap  size\_t start : 설정하려는 bit의 시작 index  size\_t cnt : 설정하려는 bit의 수  bool value : 설정하려는 값 |
| **Return** | None |
| **Function** | bitmap의 start부터 cnt 수만큼 bit를 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  bitmap\_count **(**const struct bitmap **\***b**,** size\_t start**,** size\_t cnt**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수  bool value : 확인하려는 값 |
| **Return** | bitmap의 start부터 cnt 수만큼 bit들 중 값이 value인 bit들의 수 |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들 중 값이 value인 bit들의 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_contains **(**const struct bitmap **\***b**,** size\_t start**,** size\_t cnt**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수  bool value : 확인하려는 값 |
| **Return** | 값이 value인 bit가 존재하면 true, 아니면 false |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들 중 값이 value인 bit가 존재하는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_any **(**const struct bitmap **\***b**,** size\_t start**,** size\_t cnt**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수 |
| **Return** | 값이 true(1)인 bit가 존재하면 true, 아니면 false |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들 중 값이 true인 bit가 존재하는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_none **(**const struct bitmap **\***b**,** size\_t start**,** size\_t cnt**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수 |
| **Return** | 값이 true(1)인 bit가 존재하지 않으면 true, 아니면 false |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들 중 값이 true인 bit가 존재하는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_all **(**const struct bitmap **\***b**,** size\_t start**,** size\_t cnt**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수 |
| **Return** | 값이 모두 true(1)이면 true, 아니면 false |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들의 값이 모두 true인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  bitmap\_scan **(**const struct bitmap **\***b**,** size\_t start**,** size\_t cnt**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수  bool value : 확인하려는 값 |
| **Return** | start부터 cnt 수만큼 bit들 값이 value인 그룹의 시작 index |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들 값이 value인 그룹의 시작 index를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  bitmap\_scan\_and\_flip **(**struct bitmap **\***b**,** size\_t start**,** size\_t cnt**,** bool value**)** |
| **Parameter** | struct bitmap **\***b : 확인하려는 bitmap  size\_t start : 확인하려는 bit의 시작 index  size\_t cnt : 확인하려는 bit의 수  bool value : 확인하려는 값 |
| **Return** | start부터 cnt 수만큼 bit들 값이 value인 그룹의 시작 index |
| **Function** | bitmap의 start부터 cnt 수만큼 bit들 값이 value인 그룹의 시작 index를 반환한다. 이 그룹의 모든 bit들의 값을 반대 값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t  bitmap\_file\_size **(**const struct bitmap **\***b**)** |
| **Parameter** | const struct bitmap **\***b : 구하려는 bitmap |
| **Return** | file 안의 bitmap을 저장하는 데에 필요한 byte 수 |
| **Function** | file 안의 bitmap을 저장하는 데에 필요한 byte 수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_read **(**struct bitmap **\***b**,** struct file **\***file**)** |
| **Parameter** | struct bitmap **\***b : 읽으려는 bitmap  struct file **\***file : 읽으려는 file |
| **Return** | file로부터 b를 읽으면 true, 아니면 false |
| **Function** | file로부터 b를 읽는다. |

|  |  |
| --- | --- |
| **Prototype** | bool  bitmap\_write **(**const struct bitmap **\***b**,** struct file **\***file**)** |
| **Parameter** | struct bitmap **\***b : 쓰려는 bitmap  struct file **\***file : 쓰려는 file |
| **Return** | file에 b를 쓰면 true, 아니면 false |
| **Function** | file에 b를 쓴다. |

|  |  |
| --- | --- |
| **Prototype** | void  bitmap\_dump **(**const struct bitmap **\***b**)** |
| **Parameter** | const struct bitmap **\***b : 출력하려는 bitmap |
| **Return** | None |
| **Function** | bitmap을 16진수로 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap**\***  bitmap\_expand**(**struct bitmap**\*** b**,** size\_t size) |
| **Parameter** | struct bitmap**\*** b : expand 하려는 bitmap  size\_t size : expand 하려는 크기 |
| **Return** | size 크기만큼 expand한 bitmap |
| **Function** | bitmap을 size 크기만큼 expand하여 반환한다. |