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

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 교수님 |
| 학번 : | 20181617 |
| 이름 : | 김채연 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void bitmapcommand(char\* input); |
| **Parameter** | parameter로 입력받은 input을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | bitmap과 관련된 command를 처리한다. 각 input을 명령어로 case를 나누어 알맞은 함수들을 불러 명령을 처리한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmaplink\* bitmap\_find(char\* name); |
| **Parameter** | parameter로 bitmap의 이름인 name을 받는다. |
| **Return** | create된 bitmap이 연결되어있는 struct bitmaplink 타입을 return한다. name의 이름을 가진 bitmap의 bitmaplink를 return한다. |
| **Function** | name의 이름을 가지고 있는 bitmap을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void hashcommand(char\* input); |
| **Parameter** | parameter로 입력받은 input을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | hash table과 관련된 command를 처리한다. 각 input을 명령어로 case를 나누어 알맞은 함수들을 불러 명령을 처리한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hashlink\* hash\_namefind(char\* name); |
| **Parameter** | parameter로 hashtable의 이름인 name을 받는다. |
| **Return** | create된 hashtable이 연결되어있는 struct hashlink 타입을 return한다. name의 이름을 가진 hashtable의 hashlink를 return한다. |
| **Function** | name의 이름을 가지고 있는 hashtable을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void listcommand(char\*input); |
| **Parameter** | parameter로 입력받은 input을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | hash table과 관련된 command를 처리한다. 각 input을 명령어로 case를 나누어 알맞은 함수들을 불러 명령을 처리한다. |

|  |  |
| --- | --- |
| **Prototype** | struct listlink\* list\_find(char\* name); |
| **Parameter** | parameter로 list의 이름인 name을 받는다. |
| **Return** | create된 list가 연결되어있는 struct listlink 타입을 return한다. name의 이름을 가진 list의 listlink를 return한다. |
| **Function** | name의 이름을 가지고 있는 list를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void createcommand(char \*input); |
| **Parameter** | parameter로 입력받은 input을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | create command를 처리한다. input에서 자료구조 이름을 뽑아 해당 이름의 자료구조를 create한다. 각 자료구조는 각각 linkedlist로 연결된다. |

|  |  |
| --- | --- |
| **Prototype** | void dumpcommand(char \* input); |
| **Parameter** | parameter로 입력받은 input을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | dump command를 처리한다. input에서 자료구조 이름을 뽑아 해당 이름의 자료구조를 dump하는 함수를 불러 처리한다. |

|  |  |
| --- | --- |
| **Prototype** | void deletecommand(char\* input); |
| **Parameter** | parameter로 입력받은 input을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | delete command를 처리한다. input에서 자료구조 이름을 뽑아 해당 이름의 자료구조를 찾아 제거하고 free한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump2(struct bitmap\* bitmap); |
| **Parameter** | dump할 bitmap인 bitmap을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 bitmap의 전체 bit를 dump한다. true는 1, false는 1로 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_dump(struct hash\* hash); |
| **Parameter** | dump할 hashtable인 hash를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 hash 전체를 dump한다. hash\_entry를 이용하여 hash의 data를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_dump(struct list\* list); |
| **Parameter** | dump할 list인 list를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list 전체를 dump한다. list\_entry를 이용하여 list의 data를 출력한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head (struct list\_elem \*elem); |
| **Parameter** | list들 사이의 prev, next 링크를 저장하고 있는 struct인 list\_elem을 parameter로 받는다. |
| **Return** | bool type을 return한다. elem이 list의 head라면 true, head가 아니라면 false를 return한다. |
| **Function** | parameter로 받은 elem이 list의 head 인지 검사한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior (struct list\_elem \*elem); |
| **Parameter** | list들 사이의 prev, next 링크를 저장하고 있는 struct인 list\_elem을 parameter로 받는다. |
| **Return** | bool type을 return한다. elem이 list의 내부에 있다면(head나 tail이 아니면) true, 내부에 없다면 false를 return한다. |
| **Function** | parameter로 받은 elem이 list의 내부에 있는지 검사한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail (struct list\_elem \*elem); |
| **Parameter** | list들 사이의 prev, next 링크를 저장하고 있는 struct인 list\_elem을 parameter로 받는다. |
| **Return** | bool type을 return한다. elem이 list의 tail이라면 true, tail이 아니라면 false를 return한다. |
| **Function** | parameter로 받은 elem이 list의 tail인지 검사한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list를 초기화한다. list의 head와 tail을 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap (struct list\_elem \*a, struct list\_elem \*b); |
| **Parameter** | list의 prev, next링크를 저장하고 있는 struct인 list\_elem 두개를 parameter로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list\_elem 두개를 list내에서 서로 swap한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list의 내부를 랜덤으로 shuffle 해준다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less (const struct list\_elem\*a, const struct list\_elem\*b, void \*aux); |
| **Parameter** | list의 prev, next링크를 저장하고 있는 struct인 list\_elem 두개를 parameter로 받는다. aux는 parameter로 받지만 함수내에서 사용되지 않는다. |
| **Return** | bool type을 return한다. a의 data가 b의 data보다 작으면 true, 크거나 같으면 false를 return한다. |
| **Function** | parameter로 받은 두개의 list\_elem을 가지고 있는 list\_item을 구해 각 data값을 비교하여 a가 b보다 작은 값을 가지고 있는지 검사한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list의 시작 list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 첫번째 시작 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*elem); |
| **Parameter** | list들 사이의 prev, next 링크를 저장하고 있는 struct인 list\_elem을 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list내에서 elem의 다음 list\_elem을 return한다. |
| **Function** | parameter로 받은 lelem의 다음 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list내에서 끝 list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 마지막 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list 내에서 오른쪽에서 시작하는(끝에서 첫번째인) list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 마지막인 tail의 전 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*elem); |
| **Parameter** | list들 사이의 prev, next 링크를 저장하고 있는 struct인 list\_elem을 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list 내에서 elem의 이전 list\_elem을 return한다. |
| **Function** | parameter로 받은 elem의 이전 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list 내에서 오른쪽에서 끝인(시작에서 첫번째인) list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 첫번째인 head의 다음 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list의 head인 list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 head list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | list들 사이의 prev, next링크를 저장하고 있는 struct인 list\_elem타입을 return한다. list의 tail인 list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 tail list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem); |
| **Parameter** | insert될 자리의 이전 list\_elem을 before로 받고, insert될 list\_elem을 elem으로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 before의 다음 자리에 elem을 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | splice될 list 자리의 이전 list\_elem을 before로 받고 splice할 list의 시작 list\_elem을 first, 끝 list\_elem을 last로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 before의 다음 자리에 다른 list의 first부터 last까지를 추가한 뒤 삭제해준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*list, struct list\_elem \*elem); |
| **Parameter** | list\_elem이 추가될 list를 list로 받고 추가할 list\_elem을 elem으로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list의 list\_begin을 찾아 그 뒤에 elem을 insert한다. list의 가장 앞에 추가되는 것이다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*list, struct list\_elem \*elem); |
| **Parameter** | list\_elem이 추가될 list를 list로 받고 추가할 list\_elem을 elem으로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list의 list\_end를 찾아 그 뒤에 elem을 insert한다. list의 가장 마지막에 추가되는 것이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*elem); |
| **Parameter** | 제거할 list\_elem을 elem으로 받는다. |
| **Return** | list\_elem 타입을 return한다. 제거된 elem의 다음 list\_elem을 return한다. |
| **Function** | parameter로 받은 elem을 list에서 제거를 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*list); |
| **Parameter** | pop을 실행할 list를 list로 받는다. |
| **Return** | list\_elem 타입으로 pop된 list\_elem을 return한다. |
| **Function** | parameter로 받은 list에서 list\_front를 찾은뒤 front를 제거한다. list의 가장 앞을 pop하는 것이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*list); |
| **Parameter** | pop을 실행할 list를 list로 받는다. |
| **Return** | list\_elem 타입으로 pop된 list\_elem을 return한다. |
| **Function** | parameter로 받은 list에서 list\_back을 찾은 뒤 back을 제거한다. list의 가장 뒤를 pop하는 것이다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*list); |
| **Parameter** | 가장 앞 list\_elem을 찾을 list를 list로 받는다. |
| **Return** | list\_elem 타입으로 list의 front list\_elem을 return한다. |
| **Function** | parameter로 받은 list에서 가장 앞인 head의 다음 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*list); |
| **Parameter** | 가장 뒤 list\_elem을 찾을 list를 list로 받는다. |
| **Return** | list\_elem 타입으로 list의 back list\_elem을 return한다. |
| **Function** | parameter로 받은 list의 가장 뒤인 tail의 이전 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*list); |
| **Parameter** | size를 계산할 list를 list로 받는다. |
| **Return** | size\_t 타입으로 list의 개수를 return한다. |
| **Function** | parameter로 받은 list의 list\_begin에서 list\_end까지의 list\_elem 개수를 센다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*list); |
| **Parameter** | list의 head와 tail정보를 가지고 있는 struct인 list를 parameter로 받는다. |
| **Return** | bool type으로 list가 비어있으면 true, 비어있지 않으면 false를 return한다. |
| **Function** | parameter로 받은 list가 비어있는지 확인한다. list\_begin 과 list\_end가 같으면 그 사이 list가 없다는 뜻이므로 비어있다는 것을 알 수 있다. |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b); |
| **Parameter** | swap할 list\_elem 의 주소값으로 a와 b 두개를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list\_elem a와 b를 서로 swap 해준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*list); |
| **Parameter** | list 순서를 reverse할 list를 list로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter고 받은 list내의 순서를 reverse한다. swap 함수를 이용하여 앞, 뒤 위치를 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux); |
| **Parameter** | sort되어있는지 확인할 list의 시작 list\_elem을 a, 끝 list\_elem을 b로 받고 a와 b사이의 list\_elem들의 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈뿐 역할이 없다. |
| **Return** | bool 타입으로 a와 b사이가 작은것부터 순서대로 정렬되어 있으면 true, 그렇지 않다면 false를 return한다. |
| **Function** | a와 b사이를 less함수를 이용하여 순서대로 정렬되어 있는지 확인한다. a를 b까지 list\_next로 옮기며 a와 a의 prev를 less함수로 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem \*find\_end\_of\_run (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux); |
| **Parameter** | 함수 기능을 실행할 시작 list\_elem을 a, 끝을 b로 받고 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | list내에서 a부터 b까지 순서대로 정렬되어 있지 않는 첫번째 list\_elem을 return한다. 정렬되어있다면 b가 return된다. |
| **Function** | a에서 b까지 less 함수를 이용하여 순서대로 정렬되어 있는지 확인한 후 정렬되어 있지 않다면 첫번째 해당 list\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | static void inplace\_merge (struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux); |
| **Parameter** | 함수 기능을 실행할 시작 list\_elem을 a0, 중간을 a1b0, 끝을 b1로 받고 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | a0에서 a1b0까지 사이에 a1b0에서 b1까지를 넣는다. 이때 이 list들은 모두 작은 순으로 정렬되어 있어야 한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux); |
| **Parameter** | sorting할 list를 list로, 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list 내부를 작은순으로 정렬한다.  find\_end\_of\_run, implace\_merge의 함수를 사용한다. 이 함수들에 대한 설명은 위에 했다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux); |
| **Parameter** | insert될 list를 list로받고 insert할 list\_elem을 elem으로, 대소비교를 할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 elem을 list에 insert하는데 작은 순서대로 정렬되도록 insert한다. 즉 list내에서 elem보다 큰 list\_elem의 앞에 insert한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux); |
| **Parameter** | 함수기능을 실행할 list를 list로 받고 따로 복제하여 저장할 list를 duplicate로 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 list의 중복을 제거한다. 이때 duplicate list에 중복제거된 list\_elem들을 따로 순서대로 저장해준다. duplicate 로 NULL이 넘어왔다면 생략한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux); |
| **Parameter** | 함수기능을 실행할 list를 list로 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | list\_elem 타입을 return한다. list내에서 가장 큰 data를 가지고 있는 list\_elem을 return한다. |
| **Function** | list 내에서 가장 큰 data값을 가지고 있는 list\_elem을 less 함수를 이용하여 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux); |
| **Parameter** | 함수기능을 실행할 list를 list로 대소를 비교할 함수를 less로 받는다. aux는 parameter로 받지만 함수내에서 less함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | list\_elem 타입을 return한다. list내에서 가장 작은 data를 가지고 있는 list\_elem을 return한다. |
| **Function** | list 내에서 가장 작은 data값을 가지고 있는 list\_elem을 less 함수를 이용하여 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | list\_entry(LIST\_ELEM, STRUCT, MEMBER); |
| **Parameter** | 전환시킬 list\_elem을 첫번째로, 전환될 data type 명을 두번째로, data type내 list\_elem의 명을 세번째로 받는다. |
| **Return** | list\_elem을 입력된 data type으로 전환하여 return해준다. |
| **Function** | 주로 해당 list\_elem을 member로 가지고 있는 list\_item을 찾는 기능을 한다. 링크를 가지고 list\_elem의 data값을 알 수 있다. |

**Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux); |
| **Parameter** | 초기화할 hash를 h로, h의 hash를 hash로, h의 less를 less로, h의 aux를 aux로 받는다. |
| **Return** | bool type을 return한다. 성공적으로 초기화하면 true를 그렇지 않다면 false를 return한다. |
| **Function** | parameter로 받은 hash인 h를 초기화한다. struct hash 내의 member 값들을 모두 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int i); |
| **Parameter** | 변환할 int 값을 i로 받는다. |
| **Return** | unsigned type을 return한다. 변환된 결과 hash\_bytes를 통해 return한다. |
| **Function** | parameter로 받은 i를 하나의 값에 집중되지 않도록 적당한 값으로 변환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_function(const struct hash\_elem \*e, void \*aux); |
| **Parameter** | data값을 구할 hash\_elem을 e로 받는다. aux는 parameter로 받지만 역할이 없다. |
| **Return** | unsigned type으로 e의 data값을 hash\_int를 통해 return해준다. |
| **Function** | parameter로 받은 i를 hash\_entry를 사용하여 변환한 뒤 i의 data값을 구한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_action\_square (struct hash\_elem\*e , void \*aux); |
| **Parameter** | 함수기능을 수행할 hash\_elem을 e로 받는다. aux는 parameter로 받지만 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 e를 hash\_entry를 사용하여 변환한 뒤 data값을 구하여 제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_action\_triple(struct hash\_elem\*e, void \*aux); |
| **Parameter** | 함수기능을 수행할 hash\_elem을 e로 받는다. aux는 parameter로 받지만 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 e를 hash\_entry를 사용하여 변환한 뒤 data값을 구하여 세제곱한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less (const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux); |
| **Parameter** | 함수기능을 수행할 hash\_elem을 a, b로 두개 받는다. aux는 parameter로 받지만 함수내에서 hash\_function 함수의 인자로 들어갈 뿐 역할이 없다. |
| **Return** | bool type으로 a와 b의 데이터값을 비교하여 a가 b보다 작으면 true, 같거나 크면 false를 return한다. |
| **Function** | parameter로 받은 hash\_elem인 a와 b의 대소비교를 하여 결과를 return해준다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_free (struct hash\_elem\* e, void \*aux); |
| **Parameter** | free할 hash\_elem을 e로 받는다. aux는 parameter로 받지만 역할이 없다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 e를 member로 가지는 hash\_elem2를 구하여 hash\_elem2를 free한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor); |
| **Parameter** | 함수기능을 수행할 hash로 h를 받고 hash\_action\_func로 destructor를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 h를 clear한다. h내부의 list를 모두 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor); |
| **Parameter** | 함수기능을 수행할 hash로 h를 받고 hash\_action\_func로 destructor를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 h를 clear한뒤 free하여 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new); |
| **Parameter** | insert될 hash를 h로, insert할 hash\_elem을 new로 받는다. |
| **Return** | h 내부에 이미 new와 같은 hash\_elem이 있으면 그것을 return하고 없다면 null을 return한다. |
| **Function** | parameter로 받은 h내부에 알맞은 bucket에 new를 insert한다. 이미 존재한다면 insert하지 않는다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*new); |
| **Parameter** | replace될 hash를 h로, replace할 hash\_elem을 new로 받는다. |
| **Return** | hash\_elem 타입으로 new가 새로 삽입되면서 제거된 hash\_elem을 return한다. |
| **Function** | parameter로 받은 h 내부에 new를 삽입하는데 이미 존재하는 hash\_elem을 제거하고 new를 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | hash\_elem을 찾을 hash를 h로, 찾을 hash\_elem을 e로 받는다. |
| **Return** | hash\_elem 타입으로 e를 h내부에서 찾으면 e를 return하고 없으면 null을 return한다. |
| **Function** | parameter로 받은 h 내부에 e가 존재하는지 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | 함수기능을 수행할 hash를 h로, 제거할 hash\_elem을 e로 받는다. |
| **Return** | hash\_elem 타입으로 e가 h내부에 있으면 e를 return하고 없으면 null을 return한다. |
| **Function** | parameter로 받은 h내부에 e가 존재하는지 찾은 뒤 존재한다면 h에서 e를 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action); |
| **Parameter** | 함수기능을 수행할 hash를 h로 hash action func로 action을 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 h 내부의 모든 hash\_elem에 대해서 acton을 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h); |
| **Parameter** | 초기화할 hash\_iterator로 i를, i의 hash로 h를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 i의 iterator를 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i); |
| **Parameter** | 함수기능을 수행할 hash\_iterator로 i를 받는다. |
| **Return** | hash\_elem 타입으로 다음 hash\_elem를 return하고 다음 hash\_elem이 없다면 null을 return한다. |
| **Function** | parameter로 받은 i의 다음 hash\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i); |
| **Parameter** | 함수기능을 수행할 hash\_iterator로 i를 받는다. |
| **Return** | hash\_elem 타입으로 현재 hash\_elem을 return하고 hash table의 끝이라면 null을 return한다. |
| **Function** | parameter로 받은 i의 현재 hash\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h); |
| **Parameter** | hash\_elem의 개수를 찾을 hash로 h를 받는다. |
| **Return** | size\_t 타입으로 현재 hash의 hash\_elem의 수를 return한다. |
| **Function** | parameter로 받은 h의 hash\_elem 개수를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h); |
| **Parameter** | 함수 기능을 수행할 hash로 h를 받는다. |
| **Return** | bool type으로 hash내부에 hash\_elem이 없다면 true를, 있다면 false를 return한다. |
| **Function** | parameter로 받은 h의 내부가 비어있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*buf\_, size\_t size); |
| **Parameter** | 함수기능을 수행할 buffer로 buf를, size\_t로 size를 받는다. |
| **Return** | unsigned 타입으로 hash value를 return한다. |
| **Function** | parameter로 받은 buf 내부의 특정 size bytes만큼 hash value를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_); |
| **Parameter** | 함수 기능을 수행할 문자열로 s를 받는다. |
| **Return** | unsigned 타입으로 hash value를 return한다. |
| **Function** | parameter로 받은 s의 hash value를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int i); |
| **Parameter** | 함수기능을 수행할 int 값을 i로 받는다. |
| **Return** | unsigned type을 return한다. 변환된 결과 hash\_bytes를 통해 return한다. |
| **Function** | parameter로 받은 i의 hash value를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct list \*find\_bucket (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | bucket을 찾을 hash로 h를, hash\_elem을 e로 받는다. |
| **Return** | static struct list타입으로 e가 속한 bucket을 return한다. |
| **Function** | parameter로 받은 h 내부에서 e가 속해있는 bucket을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \* find\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e); |
| **Parameter** | hash\_elem을 찾을 hash로 h, bucket으로 bucket, hash\_elem을 e로 받는다. |
| **Return** | static struct hash\_elem 타입으로 elem이 찾아지면 그것을 return하고 없으면 null을 return한다. |
| **Function** | parameter로 받은 h 내부의 bucket에서 e와 같은 hash\_elem을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit (size\_t x); |
| **Parameter** | 함수기능을 수행할 size\_t로 x를 받는다. |
| **Return** | static inline size\_t로 결과 x를 return한다. |
| **Function** | 1로 설정되어있는 x의 최하위비트를 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t is\_power\_of\_2 (size\_t x); |
| **Parameter** | 함수기능을 수행할 size\_t로 x를 받는다. |
| **Return** | x가 2의 거듭제곱이면 true를 return하고 이외의 경우는 false를 return한다. |
| **Function** | parameter로 받은 x가 2의 거듭 제곱인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \*h); |
| **Parameter** | rehash할 hash를 h로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | ideal하게 parameter로 받은 h의 bucket수를 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e); |
| **Parameter** | elem을 추가할 hash를 h, bucket을 bucket, 추가할 elem을 e로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 h의 bucket내부에 hash\_elem으로 e를 추가한다. |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \*h, struct hash\_elem \*e); |
| **Parameter** | elem을 제거할 hash를 h, 제거할 elem을 e로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 h에서 hash\_elem으로 e를 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | hash\_entry(HASH\_ELEM, STRUCT, MEMBER) |
| **Parameter** | 전환시킬 hash\_elem을 첫번째로, 전환될 data type 명을 두번째로, data type내 hash\_elem의 명을 세번째로 받는다. |
| **Return** | hash\_elem을 입력된 data type으로 전환하여 return해준다. |
| **Function** | 주로 해당 hash\_elem을 member로 가지고 있는 hash\_elem2를 찾는 기능을 한다. 링크를 가지고 hash\_elem의 data값을 알 수 있다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx); |
| **Parameter** | parameter로 해당 index를 찾을 bit\_idx를 받는다. |
| **Return** | static inline size\_t 타입으로 element의 index를 return한다. |
| **Function** | parameter로 받은 bit\_idx 를 가지고있는 element의 index를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) ; |
| **Parameter** | parameter로 elem type을 구할 bit\_idx를 받는다. |
| **Return** | static inline elem\_타입으로 bit\_idx의 bit가 1인 elem type을 return한다. |
| **Function** | parameter로 받은 bit\_idx에 해당하는 bit가 1인 elem\_type을 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt); |
| **Parameter** | element의 수를 구할 bit\_cnt를 받는다. |
| **Return** | static inline size\_t 타입으로 bit\_cnt 의 bits에 필요한 element 수를 return한다. |
| **Function** | parameter로 받은 bit\_cnt bits에 필요한 element의 수를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt); |
| **Parameter** | bytes의 수를 구할 bit\_cnt를 받는다. |
| **Return** | static inline size\_t 타입으로 bit\_cnt의 bits에 필요한 byte의 수를 return한다. |
| **Function** | paramenter로 받은 bit\_cnt bits에 필요한 byte의 수를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand(struct bitmap \*bitmap, int size); |
| **Parameter** | expand할 bitmap을 bitmap으로, expand할 크기를 size로 받는다. |
| **Return** | struct bitmap타입으로 expand된 bitmap을 return한다. |
| **Function** | parameter로 받은 bitmap의 기존 사이즈에 size를 더해 expand한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b); |
| **Parameter** | 함수 기능을 수행할 bitmap으로 b를 받는다. |
| **Return** | static inline elem\_type으로 bit mask를 return한다. |
| **Function** | parameter로 받은 b의 실제 사용된 마지막 요쇼의 bits를 1로 설정하고 나머지는 0으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | 만들 bitmap의 bit\_cnt인 bit\_cnt를 받는다. |
| **Return** | struct bitmap타입으로 만들어진 bitmap을 return한다. 만드는데 실패하면 NULL을 return한다. |
| **Function** | parameter로 받은 bit\_cnt만큼의 bit를가지는 bitmap을 만들고 초기화한다. 이때 bits는 모두 false로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size ); |
| **Parameter** | 만들 bitmap의 bit\_cnt를 bit\_cnt로, buffer block으로 block으로, buffer block의 크기로 block\_size로 받는다. |
| **Return** | struct bitmap 타입으로 만들어진 bitmap을 return한다. |
| **Function** | parameter로 받은 bit\_cnt만큼의 bit를 가지는 bitmap을 만들고 초기화한다. 이때 동적할당을 이용하지 않고 이미 할당되어있는 block을 이용한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt); |
| **Parameter** | buffersize를 계산할 bit\_cnt를 parameter로 받는다. |
| **Return** | size\_t 타입으로 bit\_cnt만큼의 bit를 가지는 bitmap을 만들기위한 buffer size를 return한다. |
| **Function** | parameter로 받은 bit\_cnt만큼의 bit를 가지는 bitmap을 만들기위한 buffer size를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b); |
| **Parameter** | 제거할 bitmap인 b를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 b를 모두 free하고 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b); |
| **Parameter** | bit의 수를 셀 bitmap인 b를 받는다. |
| **Return** | size\_t 타입으로 b의 bits 개수를 return한다. |
| **Function** | parameter로 받은 b의 bits 개수를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool value); |
| **Parameter** | 값을 설정할 bitmap으로 bit을, 설정할 index로 idx, 설정할 값으로 value를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 주어진 b의 idx번째에 value로 값을 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx); |
| **Parameter** | 값을 1로 설정할 bitmap을 bit, 설정할 index로 idx를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 주어진 b의 idx번째를 true로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx); |
| **Parameter** | 값을 0으로 설정할 bitmap을 bit, 설정할 index로 idx를 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 주어진 b의 idx번째를 false로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx); |
| **Parameter** | flip할 bitmap을 b, flip할 index를 idx로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 주어진 b의 idx번째의 bit를 toggle한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*b, size\_t idx); |
| **Parameter** | 값을 구할 bitmap을 b, 해당 index를 idx로 받는다. |
| **Return** | bool 타입으로 구한 bit 값을 return한다. |
| **Function** | parameter로 주어진 b의 idx번째의 bit value를 구한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value); |
| **Parameter** | 값을 설정할 bitmap을 b, 값을 value로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 주어진 b의 전체 bit를 모두 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value); |
| **Parameter** | 값을 설정할 bitmap을 b, 설정할 시작 index를 start, 개수를 cnt, 값을 value로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 주어진 b의 index start번째부터 cnt개수만큼 value로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value); |
| **Parameter** | 특정 값으로 설정되어 있는 bit 개수를 구할 bitmap을 b, 찾을 시작 index를 start, 개수를 cnt, 값을 value로 받는다. |
| **Return** | size\_t 타입으로 start부터 start+cnt 사이에 value로 설정되어있는 bit의 개수를 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 start+cnt사이에 bit 값이 value로 설정되어있는 bit의 개수를 센다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value); |
| **Parameter** | value가 있는지 확인할 bitmap을 b, 찾을 시작 index를 start, 개수를 cnt, 값을 value로 받는다. |
| **Return** | bool type으로 start부터 start+cnt사이에 value가 있으면 true, 없으면 false를 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 start+cnt사이에 bit 값이 value로 설정되어 있는 bit가 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt); |
| **Parameter** | true로 설정된 bit가 있는지 확인할 bitmap을 b, 찾을 시작 index를 start, 개수를 cnt로 받는다. |
| **Return** | bool type으로 start부터 start+cnt사이에 1이 있으면 true, 없으면 false를 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 start+cnt사이에 bit 값이 true로 설정되어 있는 bit가 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt); |
| **Parameter** | false로 설정된 bit가 있는지 확인할 bitmap을 b, 찾을 시작 index를 start, 개수를 cnt로 받는다. |
| **Return** | bool type으로 start부터 start+cnt사이에 0이 있으면 true, 없으면 false를 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 start+cnt사이에 bit 값이 false로 설정되어 있는 bit가 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt); |
| **Parameter** | 모든 bit가 true로 설정되어 있는지 확인할 bitmap을 b, 찾을 시작 index를 start, 개수를 cnt로 받는다. |
| **Return** | bool type으로 start부터 start+cnt사이가 모두 1이면 true, 아니면 false를 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 start+cnt사이에 bit 값이 모두 true로 설정되어 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value); |
| **Parameter** | 함수기능을 수행할 bitmap을 b, 시작 index를 start, 개수를 cnt, 값을 value로 받는다. |
| **Return** | size\_t 타입으로 start부터 cnt개수만큼 연속된 모든 bit가 value라면 시작 index를 return하고 없다면 BITMAP\_ERROR를 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 cnt개의 연속된 모든 bit의 값이 value로 설정되어 있는 그룹이 있는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value); |
| **Parameter** | 함수기능을 수행할 bitmap을 b, 시작 index를 start, 개수를 cnt, 값을 value로 받는다. |
| **Return** | size\_t 타입으로 start부터 cnt개수만큼 연속된 모든 bit가 value라면 시작 index를 return하고 없다면 BITMAP\_ERROR를 return, cnt가 0이라면 0을 return한다. |
| **Function** | parameter로 주어진 b의 index start번째부터 cnt개의 연속된 모든 bit의 값이 value로 설정되어 있는 그룹이 있는지 확인한다. 만약 있다면 해당 그룹의 bit를 모두 toggle한다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b); |
| **Parameter** | dump할 bitmap을 b로 받는다. |
| **Return** | void 타입으로 함수의 기능만 수행할 뿐 다른 값을 return하지 않는다. |
| **Function** | parameter로 받은 b의 내용을 16인수로 변환하여 dump한다. |