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

|  |  |
| --- | --- |
| 담당 교수 : | 문의현 |
| 학번 : | 20181536 |
| 이름 : | 엄석훈 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | int main(void) |
| **Parameter** | None |
| **Return** | Return 0 if program quit normally. |
| **Function** | Read command and execute function |

|  |  |
| --- | --- |
| **Prototype** | void create\_func(int \*status) |
| **Parameter** | Kind of testing data structure |
| **Return** | None |
| **Function** | Read name and create new bitmap or hash or list with that name. |

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_func(int \*status) |
| **Parameter** | Kind of testing data structure |
| **Return** | None |
| **Function** | Read name and print every data in that data structure. |

|  |  |
| --- | --- |
| **Prototype** | void delete\_func(int \*status) |
| **Parameter** | Kind of testing data structure |
| **Return** | None |
| **Function** | Read name and delete that data structure. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_func(char \*text) |
| **Parameter** | Command of testing feature |
| **Return** | None |
| **Function** | Execute testing function about command |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\_node \*find\_bitmap\_with\_name(char \*name) |
| **Parameter** | Name of bitmap\_node |
| **Return** | bitmap\_node that matches name. |
| **Function** | Find bitmap\_node that matches parameter. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Read data and mark it in bitmap |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_all\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_all function and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_any\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_any function and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_contains\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_contains function and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_count\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_count function and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_dump function |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_expand\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_expand fuction |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_set\_all fuction |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_flip fuction |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_none\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_none fuction and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_reset fuction |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_scan\_and\_flip\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_scan\_and\_flip fuction and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_scan\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_scan fuction and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_multiple fuction |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_set fuction |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_size\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_size fuction and print result |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_test\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_test fuction and print result |

|  |  |
| --- | --- |
| **Prototype** | void hash\_func(char \*text) |
| **Parameter** | Command of testing feature |
| **Return** | None |
| **Function** | Execute testing function about command |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_node \*find\_hash\_with\_name(char \*name) |
| **Parameter** | Name of hash\_node |
| **Return** | hash\_node that matches name. |
| **Function** | Find hash\_node that matches parameter. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_insert\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute bitmap\_insert fuction and make new hash\_item |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_apply function with command |

|  |  |
| --- | --- |
| **Prototype** | void hash\_delete\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_delete function |

|  |  |
| --- | --- |
| **Prototype** | void hash\_empty\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_empty function and print result |

|  |  |
| --- | --- |
| **Prototype** | void hash\_size\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_size function and print result |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_clear function |

|  |  |
| --- | --- |
| **Prototype** | void hash\_find\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_find function and print result |

|  |  |
| --- | --- |
| **Prototype** | void hash\_replace\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute hash\_replace function |

|  |  |
| --- | --- |
| **Prototype** | void list\_func(char \*text) |
| **Parameter** | Command of testing feature |
| **Return** | None |
| **Function** | Execute testing function about command |

|  |  |
| --- | --- |
| **Prototype** | struct list\_node \*find\_list\_with\_name(char \*name) |
| **Parameter** | Name of list\_node |
| **Return** | list\_node that matches name. |
| **Function** | Find hash\_node that matches parameter. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_push\_front function |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_push\_back function |

|  |  |
| --- | --- |
| **Prototype** | void list\_front\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_front function and print result |

|  |  |
| --- | --- |
| **Prototype** | void list\_back\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_back function and print result |

|  |  |
| --- | --- |
| **Prototype** | void list\_pop\_front\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_pop\_front function |

|  |  |
| --- | --- |
| **Prototype** | void list\_pop\_back\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_pop\_back function |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_insert function |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_insert\_ordered function |

|  |  |
| --- | --- |
| **Prototype** | void list\_empty\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_empty function and print result |

|  |  |
| --- | --- |
| **Prototype** | void list\_size\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_size function and print result |

|  |  |
| --- | --- |
| **Prototype** | void list\_max\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_max function and print result |

|  |  |
| --- | --- |
| **Prototype** | void list\_min\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_min function and print result |

|  |  |
| --- | --- |
| **Prototype** | void list\_remove\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_remove function |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_reverse function |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_shuffle function |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_sort function |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_splice function |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_swap function |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique\_func(void) |
| **Parameter** | None |
| **Return** | None |
| **Function** | Execute list\_unique function |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_init(struct list \*list) |
| **Parameter** | LIST to initializes |
| **Return** | None |
| **Function** | Initializes LIST as an empty list. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*list) |
| **Parameter** | LIST to find beginning |
| **Return** | Beginning of LIST element |
| **Function** | Find LIST beginning element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*elem) |
| **Parameter** | LIST element to find next |
| **Return** | Next element in the LIST |
| **Function** | Find next LIST element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*list) |
| **Parameter** | LIST to find end |
| **Return** | End of LIST element |
| **Function** | Find LIST end element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*list) |
| **Parameter** | LIST to find reverse beginning |
| **Return** | Reverse beginning of LIST element |
| **Function** | Find LIST reverse beginning element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*elem) |
| **Parameter** | LIST element to find pervious |
| **Return** | Previous element in the LIST |
| **Function** | Find previous LIST element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*list) |
| **Parameter** | LIST to find reverse end |
| **Return** | Reverse end of LIST element |
| **Function** | Find LIST reverse end element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*list) |
| **Parameter** | LIST to find head of LIST |
| **Return** | Head of LIST |
| **Function** | Find LIST head element |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*list) |
| **Parameter** | LIST to find tail of LIST |
| **Return** | Tail of LIST |
| **Function** | Find LIST tail element |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| **Parameter** | Target position to insert and list\_elem to insert |
| **Return** | None |
| **Function** | Insert ELEM before BEFORE |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| **Parameter** | Target point to splice before, first, last |
| **Return** | None |
| **Function** | Removes elements FIRST though LAST then inserts before BEFORE |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*) |
| **Parameter** | LIST\_ELEM to push and target LIST |
| **Return** | None |
| **Function** | Insert ELEM at the beginning of LIST |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*) |
| **Parameter** | LIST\_ELEM to push and target LIST |
| **Return** | None |
| **Function** | Insert ELEM at the end of LIST |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*) |
| **Parameter** | LIST\_ELEM to remove |
| **Return** | LIST\_ELEM that was removed |
| **Function** | Remove LIST\_ELEM from the LIST |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*) |
| **Parameter** | Target LIST to pop from the front |
| **Return** | Poped LIST\_ELEM |
| **Function** | Remove one LIST\_ELEM from the front of the LIST |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*) |
| **Parameter** | Target LIST to pop from the back |
| **Return** | Poped LIST\_ELEM |
| **Function** | Remove one LIST\_ELEM from the back of the LIST |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*) |
| **Parameter** | Target LIST to find front element |
| **Return** | Front element in LIST |
| **Function** | Find front element in the LIST |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*) |
| **Parameter** | Target LIST to find lack element |
| **Return** | Last element in LIST |
| **Function** | Find last element in the LIST |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*) |
| **Parameter** | Target LIST to find size |
| **Return** | Size of the LIST |
| **Function** | Find Size of the LIST |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*) |
| **Parameter** | Target LIST to check empty |
| **Return** | If LIST is empty return true else false |
| **Function** | Check LIST is empty |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*) |
| **Parameter** | Target LIST to reverse |
| **Return** | None |
| **Function** | Reverse order of the LIST |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*, list\_less\_func \*, void \*aux) |
| **Parameter** | Target LIST to sort with list\_less\_func |
| **Return** | None |
| **Function** | Sort order of the LIST by list\_less\_func |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux) |
| **Parameter** | ELEM to insert and target LIST, and list\_less\_func to check order |
| **Return** | None |
| **Function** | Put ELEM in the LIST with list\_less\_func order |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux) |
| **Parameter** | Target LIST and target duplicate LIST and list\_less\_func |
| **Return** | None |
| **Function** | Remove same value of LIST and put it to duplicates LIST |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux) |
| **Parameter** | Target List and list\_less\_func |
| **Return** | LIST\_ELEM that is maximum from the LIST |
| **Function** | Find maximum lists\_elem with list\_less\_func |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux) |
| **Parameter** | Target List and list\_less\_func |
| **Return** | LIST\_ELEM that is minimum from the LIST |
| **Function** | Find minimum lists\_elem with list\_less\_func |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | Two list elements that will be swapped |
| **Return** | None |
| **Function** | Swap two list elements in parameters |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | List that will be shuffled |
| **Return** | None |
| **Function** | Shuffle elements of LIST in the parameter |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | Two LIST\_ELEM to compare |
| **Return** | if B is larger than a return true, else return false |
| **Function** | Check which LIST\_ELEM has bigger value |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux) |
| **Parameter** | HASH to initializes, hashing function, comparing function |
| **Return** | If initialized success return true, else return false |
| **Function** | Initializes new hash |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | Target hash to clear and destructor action function |
| **Return** | None |
| **Function** | Remove all the elements in the hash |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor) |
| **Parameter** | Target hash to destory and destructor action function |
| **Return** | None |
| **Function** | Destory hash table. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new) |
| **Parameter** | New hash\_elem and target hash table |
| **Return** | If target hash\_elem’s hashed value is in the hash then return it, else NULL |
| **Function** | Inserts new hash\_elem into hash table |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | New hash\_elem and target hash table |
| **Return** | If target hash\_elem’s hashed vale is in the hash then return it, else NULL |
| **Function** | Inserts or replace new hash\_elem into hash table |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | Target hash\_elem to find and hash table |
| **Return** | Target hash\_elem |
| **Function** | Find hash\_elem in the hash table |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | Target hash\_elem to find and hash table |
| **Return** | Target hash\_elem |
| **Function** | Find hash\_elem and delete in the hash table |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*h, hash\_action\_func \*action) |
| **Parameter** | Target hash table and action function |
| **Return** | None |
| **Function** | Execute action function to hash table |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*i, struct hash \*h) |
| **Parameter** | Target hash table and iterator |
| **Return** | None |
| **Function** | Initializes iterator for hash table |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*i) |
| **Parameter** | Iterator for the hash table |
| **Return** | Next hash\_elem |
| **Function** | Find next hash\_elem to the iterator position |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*i) |
| **Parameter** | Iterator for the hash table |
| **Return** | Currunt hash\_elem |
| **Function** | Find currunt hash\_elem where iterator is. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*h) |
| **Parameter** | Target hash table to find size |
| **Return** | Size of hash table |
| **Function** | Find size of hash table |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*h) |
| **Parameter** | Target hash table to check empty |
| **Return** | If hash talbe is empty return true, else return false |
| **Function** | Check hash table is empty |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*bif, size\_t size) |
| **Parameter** | Target byte to hash with size |
| **Return** | Hashed value |
| **Function** | Hasing the byte |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*s\_) |
| **Parameter** | Target string to hash |
| **Return** | Hashed value |
| **Function** | Hashing the string |

|  |  |
| --- | --- |
| **Prototype** | unsiged hash\_int (int i) |
| **Parameter** | Target integer to hash |
| **Return** | Hashed value |
| **Function** | Hashing the integer value |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | Integer that will be hashed |
| **Return** | Hash vale of integer I |
| **Function** | Hashing by change each 32bit equally with multiply with specific value. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_hash(const struct hash\_elem \*a, void \*aux) |
| **Parameter** | Target hash\_elem to hash |
| **Return** | Hashed value of hash\_elem |
| **Function** | Hashing the value of hash\_elem |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | Two hash\_elem to compare |
| **Return** | If hashed value of b is larger than a return true, else false |
| **Function** | Compare two hash\_elem’s hashed value |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem \*a, void \*aux) |
| **Parameter** | Target hash\_elem to square |
| **Return** | None |
| **Function** | Square the value of hash\_elem |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem \*a, void \*aux) |
| **Parameter** | Target hash\_elem to triple |
| **Return** | None |
| **Function** | Triple the value of hash\_elem |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt) |
| **Parameter** | Size of new bitmap |
| **Return** | New bitmap |
| **Function** | Create new bitmap with size |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*block, size\_t block\_size) |
| **Parameter** | Size of new bitmap and |
| **Return** | New bitmap |
| **Function** | Create new bitmap with size in block\_size byte |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt) |
| **Parameter** | Size of bitmap |
| **Return** | Number of bytes required to bitmap |
| **Function** | Calculate and return number of bytes required to bitmap |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b) |
| **Parameter** | Target bitmap to destory |
| **Return** | None |
| **Function** | Destroy bitmap and freeing its storage |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*b) |
| **Parameter** | Target bitmap to know size |
| **Return** | Size of bitmap |
| **Function** | Return size of bitamp |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*b, size\_t idx, bool) |
| **Parameter** | Target bitmap, position, bool type. |
| **Return** | None |
| **Function** | Set bitmap in idx’s position to bool |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*b, size\_t idx) |
| **Parameter** | Target bitmap, position |
| **Return** | None |
| **Function** | Mark bitmap in idx’s position to 1 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*b, size\_t idx) |
| **Parameter** | Target bitmap, position |
| **Return** | None |
| **Function** | Reset bitmap in idx’s position to 0 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*b, size\_t idx) |
| **Parameter** | Target bitmap, position |
| **Return** | None |
| **Function** | Flip bitmap in idx’ position 0 to 1, 1 to 0 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| **Parameter** | Target bitmap, position |
| **Return** | Value of idx’s position in bitmap |
| **Function** | Return value of idx’s position in bitmap |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*b, bool value) |
| **Parameter** | Target bitmap, value |
| **Return** | None |
| **Function** | Set all value in bitmap to bool value |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Target bitmap, value, starting point, size |
| **Return** | None |
| **Function** | Set value in bitmap from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Target bitmap, value, starting point, size |
| **Return** | Number of bits in bitmap with same value in position |
| **Function** | Count same bits in bitmap from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Target bitmap, value, starting point, size |
| **Return** | If bitmap fill with value in start, start+size return true, else false |
| **Function** | Check bitmap fill with value from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | Target bitmap, value, starting point, size |
| **Return** | If bitmap have 1 in start, start+size return true, else false |
| **Function** | Check bitmap have 1 from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | Target bitmap, starting point, size |
| **Return** | If bitmap doesn’t have 1 in position return true, else false |
| **Function** | Check bitmap doesn’t have 1 from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | Bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt) |
| **Parameter** | Target bitmap, starting point, size |
| **Return** | If bitmap doesn’t have 0 in position return true, else false |
| **Function** | Check bitmap doesn’t have 0 from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Target bitmap, starting point, size, bool type |
| **Return** | Starting index of the first group of all have same value. |
| **Function** | Find starting index of the first group of all have same value from start to start+size |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value) |
| **Parameter** | Target bitmap, starting point, size, bool type |
| **Return** | Starting index of the first group of all have same value. |
| **Function** | Execute bitmap\_scan and flip all value in that position |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*b) |
| **Parameter** | Target bitmap |
| **Return** | Number of bytes needed to store bitmap in file |
| **Function** | Find number of bytes needed to store bitmap in file |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read (struct bitmap \*b, struct file \*file) |
| **Parameter** | Target bitmap and file |
| **Return** | Read successful then return true, else false. |
| **Function** | Read bitmap from the file |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write (const struct bitmap \*b, struct file \*file) |
| **Parameter** | Target bitmap and file |
| **Return** | Write successful then return true, else false. |
| **Function** | Write bitmap to the file |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*b) |
| **Parameter** | Target bitmap |
| **Return** | None |
| **Function** | Print all bitmap element in hexadecimal |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | Bitmap that you want to expand and the size of it |
| **Return** | Expanded bitmap if succeed, NULL if fail |
| **Function** | Expand the given BITMAP to the SIZE (backward expansion) |