**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 교수님 |
| 학번 : | 20201617 |
| 이름 : | 이상연 |
|  |  |

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | static struct list\_item \*create\_list\_item (int data) |
| **Parameter** | list\_item 의 data 에 넣을 값을 인자로 전달받는다. |
| **Return** | malloc을 통해 동적할당 된 list\_item의 포인터를 리턴한다. |
| **Function** | 메모리 동적할당을 통해 list\_item 구조체를 생성하고 인자로 받은 data를 생성한 list\_item 구조체의 data 값으로 저장한다. 그리고 이 구조체의 포인터를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \*create\_hash\_elem (int data) |
| **Parameter** | hash\_elem의 data 에 넣을 값을 인자로 전달받는다. |
| **Return** | malloc을 통해 동적할당 받은 hash\_elem의 포인터를 리턴한다. |
| **Function** | 메모리 동적할당을 통해 hash\_elem 구조체를 생성하고 인자로 받은 data를 생성한 hash\_elem 구조체의 data 값으로 저장한다. 그리고 이 구조체의 포인터를 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | int parse\_command (char \*command, char \*\*command\_ptr) |
| **Parameter** | 사용자로부터 입력받은 한 줄의 command 라인과 이를 splice 해서 저장하기 위한 char\*\* 를 인자로 받는다. |
| **Return** | 공백을 기준으로 나눠지는 명령어의 단어의 수를 리턴한다. |
| **Function** | 인자로 받은 command 는 사용자가 입력한 한 줄의 명령어이고, command\_ptr 은 char\*\* 변수이다. strchr 함수를 이용해서 command 의 마지막 줄바꿈 문자를 null 문자로 바꾸고, strtok 를 이용해서 공백(“ “)을 기준으로 단어를 command 를 파싱해서 command\_ptr 의 각 원소에 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | int parse\_num (int option, char \*buf) |
| **Parameter** | 어떤 자료구조의 명령어 인지를 나타내는 option과, 명령어 char \*를 인자로 받는다. |
| **Return** | 명령어의 대상이 되는 자료구조의 이름 뒤의 숫자를 반환한다. |
| **Function** | 인자로 전달받은 option은 0인 경우 list, hashtable 에 대한 명령어임을 의미하고, 1인 경우는 bitmap 에 대한 명령어임을 의미하는 데에 사용된다. 각 경우에 따라 알맞은 자리의 문자를 숫자로 변환해서 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void execute\_command (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 argc와 argv는 각각 명령어 문장의 공백으로 구분되는 단어의 개수와 그 단어들 각각이 저장되어 있는 char\*\* 변수이다(이 항목에서 이후의 함수들은 모두 이와 같은 파라미터 구조를 사용하기 때문에 이후에 이 부분에 대한 내용은 생략한다.). cmd\_table을 순회하며 argv[0] (명령어 문장의 첫 번째 단어)와 일치하는 문자열이 존재하는지 확인하고, 그 문자열을 찾으면 해당하는 함수를 호출하고 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_create (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | create 명령어는 모든 자료구조에서 공통적으로 사용하는 명령어이다. 함수 내부에서는 argv를 적절히 인덱싱해서 조건문을 통해 각 자료구조에 해당하는 명령어 함수를 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_dumpdata (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | dumpdata 명령어는 모든 자료구조에서 공통적으로 사용하는 명령어이다. 함수 내부에서는 argv를 적절히 인덱싱해서 조건문을 통해 각 자료구조에 해당하는 명령어 함수를 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_delete (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | delete 명령어는 모든 자료구조에서 공통적으로 사용하는 명령어이다. 함수 내부에서는 argv를 적절히 인덱싱해서 조건문을 통해 각 자료구조에 해당하는 명령어 함수를 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_push\_back (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_push\_back() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. 그리고 list\_push\_back() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_insert (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_insert() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. 그리고 list\_insert() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_swap (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_swap() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. 그리고 list\_swap() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_splice (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_splice() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_access() 함수를 호출해서 list\_splice() 함수 호출 시에 인자로 전달할 값들을 얻는다. 그리고 list\_splice() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_sort (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_sort() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_sort() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_unique (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_unique() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_unique() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_reverse (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_reverse() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_reverse() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_remove (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_remove() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_access() 함수를 호출해서 list\_remove() 함수의 인자로 넘길 값(struct list\_elem \*)을 얻어낸다. list\_remove() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_front (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_front() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_front() 함수를 호출해서 얻어낸 값을 list\_entry() 연산을 통해 struct list\_item \* 변수로 받는다. 이 변수의 data 값을 참조해서 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_back (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_back() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_back() 함수를 호출해서 얻어낸 값을 list\_entry() 연산을 통해 struct list\_item \* 변수로 받는다. 이 변수의 data 값을 참조해서 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_push\_front (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_push\_front() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. create\_list\_item() 함수를 호출해서 인자로 넘길 값을 저장한다. list\_push\_front() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_pop\_front (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_pop\_front() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_pop\_front() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_pop\_back (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_pop\_back() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_pop\_back() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_empty (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_empty() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_empty() 함수를 호출했을 때의 반환값에 따라 “true” 혹은 “false” 를 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_size (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_size() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_size() 함수를 호출하고 반환값을 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_max (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_max() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_max() 함수를 호출하고, list\_entry() 연산을 통해 struct list\_item \* 값으로 받는다. data 를 참조해서 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_min (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_min() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_min() 함수를 호출하고, list\_entry() 연산을 통해 struct list\_item \* 값으로 받는다. data 를 참조해서 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_insert\_ordered (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_insert\_ordered() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_sort() 함수를 호출해서 해당하는 list를 오름차순으로 정렬 후에 list\_insert\_ordered() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_list\_shuffle (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | list\_shuffle() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. list\_shuffle() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_replace (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_replace() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. create\_hash\_elem() 함수를 호출해서 인자로 넘길 struct hash\_elem \* 값을 얻어낸다. hash\_replace() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_insert (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_insert() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. create\_hash\_elem() 함수를 호출해서 인자로 넘길 struct hash\_elem \* 값을 얻어낸다. hash\_insert() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_find (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_find() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. create\_hash\_elem() 함수를 호출해서 인자로 넘길 struct hash\_elem \* 값을 얻어낸다. hash\_find() 함수를 호출하고 반환값이 존재하면 값(data)을 출력한다. 동적할당한 메모리를 해제하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_empty (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_empty() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. hash\_empty() 함수를 호출해서 반환값에 따라 “true” 또는 “false”를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_size (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_size() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. hash\_size() 함수를 호출하고 반환값을 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_clear (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_clear() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. hash\_clear() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_delete (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_delete() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. create\_hash\_elem() 함수를 호출해서 인자로 넘길 struct hash\_elem \* 값을 얻어낸다. hash\_delete() 함수를 호출하고 동적할당한 메모리를 해제하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_hash\_apply (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | hash\_apply() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. hash\_apply() 함수의 두 번째 함수 포인터 인자로 올 수 있는 경우는 제곱하는 함수, 세제곱하는 함수 두 가지의 경우가 있다. 그렇기 때문에 조건문으로 해당하는 함수 포인터를 할당하고, hash\_apply() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_mark (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_mark() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_mark() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_test (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_test() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_test() 를 호출하고 반환값에 따라 “true” 혹은 “false”를 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_size (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_size() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_size() 함수를 호출하고 반환값을 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_set (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_set() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_set() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_set\_multiple (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_set\_multiple() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_set\_multiple() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_set\_all (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_set\_all() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_set\_all() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_scan (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_scan() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_scan() 함수를 호출하고 반환값에 따라 “true” 혹은 “false”를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_scan\_and\_flip (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_scan\_and\_flip() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_scan\_and\_flip() 함수를 호출하고 반환값을 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_reset (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_reset() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_reset() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_none (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_none() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_none() 함수를 호출하고 반환값에 따라 “true” 혹은 “false” 를 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_flip (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_flip() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_flip() 함수를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_expand (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_expand() 함수를 호출하기 위해 argv 에서 필요한 인자들을 인덱싱해서 얻어낸다. bitmap\_expand() 함수를 호출하고 반환값을 bm[idx] 에 저장하고 종료한다(bitmap \* 를 리턴하고, 이를 다시 bm 전역변수에 저장). |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_dump (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_dump() 를 호출하기 위해 argv 에서 필요한 값들을 인덱싱해서 얻어낸다. bitmap\_dump() 를 호출하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_count (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_count() 를 호출하기 위해 argv 에서 필요한 값들을 인덱싱해서 얻어낸다. bitmap\_count() 를 호출하고 값을 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_contains (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_contains() 함수를 호출하기 위해 argv 에서 필요한 값들을 인덱싱해서 얻어낸다. bitmap\_contains() 함수를 호출하고 반환값에 따라 “true” 혹은 “false”를 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_any (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_any() 함수를 호출하기 위해 argv 에서 필요한 값들을 인덱싱해서 얻어낸다. bitmap\_any() 함수를 호출하고 반환값에 따라 “true” 혹은 “false”를 출력하고 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void cmd\_bitmap\_all (int argc, char \*\*argv) |
| **Parameter** | 명령어에서 공백으로 구분되는 단어의 개수 argc와 단어들이 저장되어 있는 char \*\*argv를 인자로 받는다. |
| **Return** | void |
| **Function** | bitmap\_all() 함수를 호출하기 위해 argv 에서 필요한 값들을 인덱싱해서 얻어낸다. bitmap\_all() 함수를 호출하고 반환값에 따라 “true” 혹은 “false”를 출력하고 종료한다. |

**Global Variables**

struct list list[MAX\_SIZE];

struct hash hash[MAX\_SIZE];

struct bitmap\* bm[MAX\_SIZE];

각 자료구조에 해당하는 명령어들을 처리하기 위해서 자료구조들을 저장할 배열들을 선언해서 사용했다.

typedef void (\*command\_func) (int, char \*\*);

struct command\_entry {

const char \*name;

command\_func func;

};

static struct command\_entry cmd\_table[] = {

// create / dumpdata / delete

{"create", cmd\_create},

{"dumpdata", cmd\_dumpdata},

{"delete", cmd\_delete},

...

};

execute\_command() 함수에서 참조하게 되는 전역변수이다. command\_entry 구조체를 새로 정의하고 이 구조체 배열을 전역변수로 선언했다. 구조체는 문자열과 command\_func 함수의 포인터를 멤버 변수로 갖는다.

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap (struct list\_elem \*a, struct list\_elem \*b); |
| **Parameter** | list 내에서 위치를 바꿀 두 개의 list\_elem \* 를 인자로 받는다. |
| **Return** | void |
| **Function** | 서로 swap 을 할 두 개의 list\_elem 구조체 포인터 변수를 인자로 인자로 받아서 내부적으로는 list\_item 구조체 포인터 값을 얻어 data 값만 서로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle (struct list \*list); |
| **Parameter** | shuffle 연산을 수행할 리스트 자료구조의 포인터를 인자로 받는다. |
| **Return** | void |
| **Function** | 이중 for 문을 통해 outer loop 에서의 고정된 위치를 기준으로 다른 원소들과 자리를 바꿀지 말지를 결정하며 섞는 방식으로 동작한다. rand() % 2 연산을 하면 결과가 0과 1만 나오기 때문에 이를 이용해서 0이면 list\_swap()을 호출하지 않고 1일 때에만 list\_swap()을 호출해서 위치를 서로 바꾼다. loop 가 끝나면 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_delete (struct list \*list); |
| **Parameter** | 삭제시킬 리스트 자료구조의 포인터를 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 list 앞에서부터 순차적으로 list\_pop\_front() 함수를 호출하며 원소를 리스트에서 뺀다. 그리고 free() 를 호출해서 할당한 메모리를 해제한다. 반복문에서 탈출하면(처음부터 끝까지 다 삭제하면) 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_print (struct list \*list); |
| **Parameter** | 원소들을 출력할 리스트 자료구조의 포인터를 인자로 받는다. |
| **Return** | void |
| **Function** | 리스트를 처음부터 끝까지 순회하며 list\_entry() 연산을 통해 data 값을 참조해서 출력한다. 모든 요소의 출력이 끝나면 종료한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_access (struct list \*list, int index); |
| **Parameter** | 특정 리스트의 특정 인덱스에 존재하는 list\_elem \* 를 얻기 위해, list \* 와 index 를 인자로 받는다. |
| **Return** | 인자로 받은 list의 index 위치에 있는 list\_elem \* 를 리턴한다. |
| **Function** | 인자로 받은 list의 index 에 있는 요소의 list\_elem \* 를 리턴한다. list\_empty(list) 가 false 인 경우(빈 리스트가 아닌 경우) index 를 체크해서 0과 list\_size(list) – 1 사이 이면 반복문을 통해 순회하며 해당 index의 요소를 리턴한다. 만약 index 값이 0보다 작은 경우엔 list\_begin(list) 를 리턴하고, list\_size(list) – 1 보다 큰 경우에는 list\_tail(list)을 리턴한다. 만약 빈 리스트인 경우에는 list\_begin(list)을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux); |
| **Parameter** | data 값을 비교할 두 list\_elem \*를 인자로 받는다. |
| **Return** | 인자로 받은 두 list\_elem \*의 data 값의 대소를 비교한 결과를 리턴한다. |
| **Function** | list\_sort(), list\_unique(), list\_max(), list\_min() 에서는 함수포인터 인자를 받는다. 이 때 인자로 넘겨줄 함수를 정의한다. 내부적으로는 list\_entry() 연산을 통해 인자로 받은 a, b의 struct list\_item \* 을 얻어 data 값에 접근해서 a\_data < b\_data 값을 리턴한다(boolean 값 리턴). |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_hash (const struct hash\_elem \*e, void \*aux); |
| **Parameter** | hash 값을 얻을 hash\_elem \* 를 인자로 받는다. |
| **Return** | 인자로 전달받은 원소의 hash 값을 리턴한다. |
| **Function** | 내부적으로 hash\_int() 해시 함수를 호출하고 이 함수의 반환값을 리턴하는 함수이다. hash\_hash\_func 함수 포인터의 값으로 사용하기 위해 정의했다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_print (struct hash \*h); |
| **Parameter** | 원소들을 출력할 해시 자료구조 hash \* 를 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 h를 순회하며 해당하는 값을 출력한다. struct hash\_iterator와 hash\_first(), hash\_next() 함수를 이용해서 순회한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_free(struct hash\_elem \*e, void \*aux); |
| **Parameter** | 해시 내부에서 삭제하고 메모리를 해제할 hash\_elem \*을 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 hash\_elem \* 을 free() 를 호출해서 할당된 메모리를 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square (struct hash\_elem \*e, void \*aux); |
| **Parameter** | data 값을 square 연산을 수행할 hash\_elem \*을 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 hash\_elem \* 의 data 값을 참조해서 제곱한 값을 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple (struct hash\_elem \*e, void \*aux); |
| **Parameter** | data 값을 triple 연산을 수행할 hash\_elem \*을 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 hash\_elem \* 의 data 값을 참조해서 세제곱한 값을 저장한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less (const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux); |
| **Parameter** | 원소의 data 값을 비교할 두 hash\_elem \*을 인자로 받는다. |
| **Return** | 대소비교 한 결과를 리턴한다. |
| **Function** | 인자로 전달받은 두 hash\_elem \* 인자 a, b의 data 값을 참조해서 a->data < b->data 의 값을 리턴한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int i); |
| **Parameter** | 해시 값을 얻은 정수 int i를 인자로 받는다. |
| **Return** | 인자로 받은 정수의 해시 값을 리턴한다. |
| **Function** | 추가적으로 구현한 해시 함수이다. 입력받은 정수의 hash 값을 리턴한다. 함수의 구현에서 사용한 magic number은 다음의 avalanche effect를 측정하는 special-multi-threaded test program을 통해 구해낸 숫자이다. <https://github.com/h2database/h2database/blob/master/h2/src/test/org/h2/test/store/CalculateHashConstant.java> |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_print (const struct bitmap \*b); |
| **Parameter** | 출력할 비트맵의 자료구조 포인터를 인자로 받는다. |
| **Return** | void |
| **Function** | 인자로 전달받은 bitmap \* b 의 값을 출력한다. bitmap\_size() 함수를 이용해서 bitmap의 사이즈를 얻고, 반복문을 통해 값을 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand (struct bitmap \*b, int size); |
| **Parameter** | expand 연산을 할 bitmap \*와 expand 시킬 size를 인자로 받는다. |
| **Return** | size 공간을 더 할당받은 새로운 비트맵 자료구조의 포인터를 리턴한다. |
| **Function** | 인자로 전달받은 size 만큼 b에서 늘린 새로운 bitmap 을 생성하고 bitmap \* 를 리턴한다. bitmap\_create() 함수를 이용해서 새로운 bitmap을 생성하고 반복문을 이용해서 원래 값을 복원한 후 리턴한다. |