**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 이영민 |
| 학번 : | 20211522 |
| 이름 : | 김정환 |
|  |  |

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | unsigned int hash\_hash(const struct hash\_elem\* e, void \*aux) |
| **Parameter** | hash\_elem 구조체의 pointer와 타입이 미정인 pointer를 전달받는다. |
| **Return** | 해시 값인 unsigned int 값을 리턴한다. |
| **Function** | hash를 생성할 때 hash값을 계산하는 hash\_int를 호출하여 해당 값을 리턴하는 함수로 hash table을 create하는 부분에서 hash\_init에 인자로 전달될 때 쓰이는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem\* a, const struct hash\_elem\* b, void \*aux) |
| **Parameter** | hash\_elem 구조체의 pointer 두 개와 타입이 미정인 pointer를 전달받는다. |
| **Return** | a에 있는 value가 더 작은지 여부인 boolean 타입 값을 리턴한다. |
| **Function** | 인자로 전달받은 두 hash\_elem에 있는 value를 비교하여 a가 더 작을 때 true를 리턴하는 함수이다. 이는 hash table을 create하는 부분에서 hash\_init에 인자로 전달되는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_del(struct hash\_elem\* h, void \*aux) |
| **Parameter** | hash\_elem 구조체의 pointer와 타입이 미정인 pointer를 전달받는다. |
| **Return** | 없다. |
| **Function** | hash\_elem 구조체를 할당한 메모리를 free로 해제하는 함수이다. 이는 hash table을 삭제할 때 hash\_destroy에 인자로 전달되는 함수이다. hash\_destroy 함수의 동작을 보면 bucket은 해제하나 내부에서 사용한 list의 elem은 인자로 전달한 함수가 사용되는 것을 볼 수 있고 이에 free가 진행되도록 하였다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_squ(struct hash\_elem \*e, void \*aux) |
| **Parameter** | hash\_elem 구조체의 pointer와 타입이 미정인 pointer를 전달받는다. |
| **Return** | 없다. |
| **Function** | 인자로 전달받은 hash\_elem의 값을 제곱한 값으로 바꿔서 저장하는 함수이다. 이는 hash\_apply 함수가 실행될 때 인자로 전달되는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_tri(struct hash\_elem \*e, void \*aux) |
| **Parameter** | hash\_elem 구조체의 pointer와 타입이 미정인 pointer를 전달받는다. |
| **Return** | 없다. |
| **Function** | 인자로 전달받은 hash\_elem의 값을 세제곱한 값으로 바꿔서 저장하는 함수이다. 이는 hash\_apply 함수가 실행될 때 인자로 전달되는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | void clear\_hash(struct hash\_elem \*e, void \*aux) |
| **Parameter** | hash\_elem 구조체의 pointer와 타입이 미정인 pointer를 전달받는다. |
| **Return** | 없다. |
| **Function** | 인자로 전달받은 hash\_elem을 제거하는 함수이다. 이는 앞과 뒤 element에서의 연결관계를 바꿔주고 인자로 받은 element는 free 처리를 하는 함수이다. hash\_clear 함수가 호출될 때 인자로 전달하는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_val\_cmp(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | list\_elem 구조체의 pointer를 2개를 각각 a, b로, 타입이 미정인 pointer를 전달받는다. |
| **Return** | list element인 a에 있는 data가 더 작은지 여부인 boolean 값을 리턴한다. |
| **Function** | 인자로 전달받은 두 list element의 값을 비교하는 함수이다. list\_elem이 속한 list\_item을 찾기 위해 list\_entry를 사용하였다. 이를 이용하여 찾은 list\_item에 있는 data를 비교하여 a가 더 작을 때 true를 리턴한다. list\_inserted\_order, list\_max, list\_min, list\_sort, list\_unique 함수가 호출될 때 인자로 전달된다. |

|  |  |
| --- | --- |
| **Prototype** | int main() |
| **Parameter** | 없다. |
| **Return** | int 타입 값을 리턴한다. |
| **Function** | 지역 변수인 크기가 100인 char 타입 array인 command에 명령을 입력받는다. 이 때 stdin으로 입력을 받으므로 scanf와 fgets에서 stdin으로부터 입력을 받는 2가지 방식을 사용하여 입력을 받았다. 이 후에 strcmp를 이용하여 비교하는 if문을 통해서 각각에 맞는 명령을 수행하도록 하는 과정을 통해서 주어진 testcase에 맞도록 메모리를 할당하고 값을 저장하거나 함수를 호출하는 등의 동작을 수행하도록 했다. 전역 변수로는 list, hash, bitmap에 대해서 해당 구조체를 15 크기의 array로 각각 가지는 list\_array, hash\_list, bitmap\_list를 선언하여 사용하였고, 이는 main 함수에서 사용하는 자료구조를 저장하는 역할을 수행했다. 이외 수정 사항으로는 list.c에 있던 list\_elem\_to\_hash\_elem를 main에서도 쓰기 위해서 list.h로 이동한 점이 있다. 또한 list에서 값 저장은 list\_item 구조체를 선언하여 사용했고 여기에는 list\_elem 하나와 int type인 data가 들어갔다. hash에서는 hash\_elem에 value라는 int 타입 변수를 추가하여 값을 저장했다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*, struct list\_elem \*); |
| **Parameter** | list\_elem 구조체의 pointer를 2개를 각각 a, b로 전달받는다. |
| **Return** | 없다. |
| **Function** | 인자로 전달받은 list의 element 2개를 서로 swap하는 함수이다. 이때 a,b가 우선 같은 경우에는 필요가 없으므로 그냥 return하는 경우를 넣어주었다. 다음으로 분리한 경우는 a, b가 서로 연결되어 있는 경우를 분리하였다. 해당 경우는 swap에서 주소값을 넣어줄 때 순서에 따라서 달라지기에 if문으로 따로 분리했다. list에서 head.next부터 값을 넣는 등의 특징이 있긴 하지만 예외 처리를 위해서 각각의 swap을 동작할 때 a, b의 앞 뒤에 있는 element가 NULL인지 확인 후에 해당 element들의 prev나 next에 a, b를 넣도록 하였다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list); |
| **Parameter** | list 구조체의 pointer를 하나 전달받는다. |
| **Return** | 없다. |
| **Function** | 인자로 전달받은 list에 있는 linked list를 무작위로 섞는 함수이다. 우선 random을 구현하기 위해서 list.c에는 time.h 헤더파일을 추가로 사용하였다. 이 후에 srand와 time을 이용하여 테이블이 실시간으로 변할 수 있도록 했다. 다음으로 list 구조체 내에는 element의 개수가 저장된 변수가 없으므로 전체 개수를 세었다. list\_size 함수가 존재하지만 size\_t 타입이 아닌 int 타입을 사용하기 위해 함수 내에서 for 문으로 세도록 하였다. random shuffle을 구현한 방식은 index를 선형으로 탐색하면서 random하게 나온 index와 swap하는 방식을 사용했다. list를 선형으로 탐색하면서 섞으면 순서가 꼬일 수 있으므로 index의 수를 올리고 swap 전에 탐색으로 element를 찾는 방식을 사용했다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i); |
| **Parameter** | int 타입의 변수 하나를 전달받는다. |
| **Return** | 해시 값인 unsigned 타입의 값을 리턴한다. |
| **Function** | 해시 값을 계산하여 리턴하는 함수이다. 해시 값을 계산하는 방식을 구해야 하는데 hash\_init에서 bucket의 개수가 4였으므로 그에 맞춰보았다. 단순히 4의 나머지로 나누는 것은 값이 기존의 탐색과 큰 차이가 없다고 생각하여 bit연산으로 섞어보기로 하였다. 따라서 각각 인자로 전달된 i를 shift 연산한 두 가지와 XOR 연산을 한 값에서 3과 and 연산을 해서 마지막 2bit로 구분하도록 하였다. 이를 return하여 규칙성은 가지면서 조금은 섞일 수 있도록 하였다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size); |
| **Parameter** | bitmap 구조체의 pointer 하나와 int 타입의 변수 하나를 전달받는다. |
| **Return** | bit 수를 확장한 bitmap 구조체의 pointer를 리턴한다. |
| **Function** | 해당 함수는 bitmap에서 bit의 개수를 확장하는 역할을 수행하는 함수이다. 우선 인자로 전달받은 기존 bitmap의 bit\_cnt에 size를 더해서 늘어날 최종 길이를 size\_t 타입의 변수에 저장한다. 그 후 bitmap\_create에 새로운 size를 전달하여 생성한 새로운 bitmap을 생성한다. 이 때 생성에 오류가 있어 NULL인 경우에는 NULL을 리턴하며 함수를 종료하도록 하고, 아닌 경우에는 기존의 bitmap에 저장되어 있던 데이터를 복사한다. 이 때 for문으로 탐색하며 bitmap의 index에 있는 값에 따라 true, false를 리턴하는 bitmap\_test가 true일 때가 true가 저장된 것이므로 그 때 bitmap\_mark로 i index에 true로 표기한다. 그 후 새로운 bitmap의 pointer를 리턴하며 함수가 종료된다. |