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

|  |  |
| --- | --- |
| 담당 교수 : | 문의현 |
| 학번 : | 20171759 |
| 이름 : | 신재현 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void do\_list(char\* operation) |
| **Parameter** | 사용자가 입력한 operation |
| **Return** | 없음 |
| **Function** | 사용자가 입력한 list 관련 동작을 구현한다. |

|  |  |
| --- | --- |
| **Prototype** | void do\_hash(char\* operation) |
| **Parameter** | 사용자가 입력한 operation |
| **Return** | 없음 |
| **Function** | 사용자가 입력한 hash 관련 동작을 구현한다. |

|  |  |
| --- | --- |
| **Prototype** | void do\_bitmap(char\* operation) |
| **Parameter** | 사용자가 입력한 operation |
| **Return** | 없음 |
| **Function** | 사용자가 입력한 bitmap 관련 동작을 구현한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | 서로 값을 바꿀 두 list\_elem의 주소 |
| **Return** | 없음 |
| **Function** | a와 b에 들어있는 값을 서로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list\* list) |
| **Parameter** | 섞을 list의 주소 |
| **Return** | 없음 |
| **Function** | 주어진 list 내부에 있는 원소들의 위치를 무작위로 섞는다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_print(struct list\* list) |
| **Parameter** | 프린트할 list의 주소 |
| **Return** | 없음 |
| **Function** | 주어진 list를 정방향으로 iteration 하면서 들어 있는 value를 stdout으로 출력한다. 각 value는 공백문자로 구분된다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_free(struct list\* list) |
| **Parameter** | 동적 해제할 list의 주소 |
| **Return** | 없음 |
| **Function** | 주어진 list를 정방향으로 iteration 하면서 모든 list node를 동적 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_compare(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | 값을 비교할 두 list\_elem 및 사용자 정의 데이터 aux |
| **Return** | a가 b보다 작으면 true, 그렇지 않으면 false |
| **Function** | 주어진 두 개의 노드의 크기를 비교한다. Sort에 사용되는 비교 함수다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned int hash\_function(const struct hash\_elem \*e, void \*aux) |
| **Parameter** | Hashing 할 데이터가 있는 hash\_elem, 사용자 정의 데이터 aux |
| **Return** | Hash 값 |
| **Function** | Hashing 할 데이터가 있는 hash\_elem을 받아서 구조체 내부에 있는 값을 hashing한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_compare(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | 값을 비교할 두 hash\_elem 및 사용자 정의 데이터 aux |
| **Return** | a가 b보다 작으면 true, 그렇지 않으면 false |
| **Function** | 주어진 두 개의 노드의 크기를 비교한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_action\_print(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 값을 출력할 hash\_elem 및 사용자 정의 데이터 aux |
| **Return** | 없음 |
| **Function** | hash action function 중, hash 데이터 출력하는 함수. hash\_apply와 함께 쓰인다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_action\_square(struct hash\_elem \*e, void\* aux) |
| **Parameter** | 값을 제곱할 hash\_elem 및 사용자 정의 데이터 aux |
| **Return** | 없음 |
| **Function** | hash\_elem의 값을 제곱한 뒤 stdout에 출력한다 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_action\_triple(struct hash\_elem \*e, void\* aux) |
| **Parameter** | 값을 세제곱할 hash\_elem 및 사용자 정의 데이터 aux |
| **Return** | 없음 |
| **Function** | hash\_elem의 값을 세제곱한 뒤 stdout에 출력한다 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int); |
| **Parameter** | Hashing 할 숫자 |
| **Return** | Hashing 된 숫자 |
| **Function** | Knuth multiplicative hash function. Target 숫자에 임의의 소수를 곱한 뒤, 소수점 이하의 부분만 취해서 2의 n승을 곱한다. 이 때, 임의의 소수는 일반적으로 황금비 숫자를 곱하는 것이 최적으로 알려져 있어서, 이를 사용했다. 또, integer overflow를 막기 위해서 마지막에 1024(2^10)을 곱한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_print (const struct bitmap \*) |
| **Parameter** | 출력 할 bitmap |
| **Return** | 없음 |
| **Function** | 전달 된 bitmap 구조체에 들어있는 데이터를 출력하되, lsb가 가장 먼저 출력되도록 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size) |
| **Parameter** | 확장 할 bitmap, 확장 할 bit 개수 |
| **Return** | 확장 된 데이터가 들어있는 구조체 |
| **Function** | 전달 된 bitmap 구조체에 들어있는 데이터를 size bit 만큼 뒤로 확장한 뒤, 새로운 구조체를 만들어서 반환한다. |