**Project #1 : MyLib**

|  |  |
| --- | --- |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | unsigned my\_hash\_int(const struct hash\_elem \*e, void \*aux) |
| **Parameter** | e: Hash element to hash, aux: Auxiliary data (unused) |
| **Return** | unsigned: Hash value of the element's data |
| **Function** | Generates a hash value for a hash table element by using hash\_int on its integer data. This is needed when creating a hash table in main.c |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux) |
| **Parameter** | a, b: Hash elements to compare, aux: Auxiliary data (unused) |
| **Return** | bool: True if a’s data is less than b’s data, false otherwise |
| **Function** | Compares two hash elements based on their integer data for sorting or ordering in the hash table. Used in main.cfor hash table operations |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem \*e, void \*aux) |
| **Parameter** | e: Hash element to modify, aux: Auxiliary data (unused) |
| **Return** | None |
| **Function** | Changes the integer value of a hash element by squaring it. This function runs when "square" is selected in main.c |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem \*e, void \*aux) |
| **Parameter** | e: Hash element to modify, aux: Auxiliary data (unused) |
| **Return** | None |
| **Function** | Changes the integer value of a hash element by cubing it. This function runs when "triple" is selected in main.c |

|  |  |
| --- | --- |
| **Prototype** | int find\_container(const char \*name) |
| **Parameter** | name: Name of the container to find |
| **Return** | int: Index of the container if found, -1 otherwise |
| **Function** | Searches for a container (list, hash, or bitmap) by name and returns its index in the global arrays. This is used in main.cfor managing different containers. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*get\_elem\_at(struct list \*list, size\_t pos) |
| **Parameter** | list: List to search, pos: Position to find |
| **Return** | struct list\_elem \*: Pointer to the element at the position, or tail if out of bounds |
| **Function** | Finds and returns the element at a given position in a list. If the position is out of range, it returns the last element. This function is used in main.c for list operations like removing or splicing elements. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | bool list\_less(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | a, b : List elements to compare, aux : Auxiliary data (unused) |
| **Return** | bool: True if data of a is less than data of b, false otherwise |
| **Function** | Compares two list elements based on their integer data. This function is used in sorting and other list operations in main.c |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | a, b: Pointer of two list elements that will be swapped |
| **Return** | None |
| **Function** | After checking whetehr two list elements are adjacent or not, swaps the positions of two elements in a list(Applying different algorithm depending on whether they are adjacent or not). This function is used in list\_shuffle |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | list: Pointer of list that will be shuffled |
| **Return** | None |
| **Function** | Randomly shuffles the elements of the list using the Fisher-Yates algorithm with using list\_swap function |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i) |
| **Parameter** | i: Integer that will be hashed |
| **Return** | unsigned: Hash value of integer i |
| **Function** | Alternative hash function for integers using bit shifting and XOR for better distribution |

|  |  |
| --- | --- |
| **Prototype** | #define hash\_entry(HASH\_ELEM, STRUCT, MEMBER) ((STRUCT \*)((char \*)(HASH\_ELEM) - offsetof(STRUCT, MEMBER))) |
| **Parameter** | HASH\_ELEM: Pointer to a hash element, STRUCT: The structure type, MEMBER: The name of the hash element inside that structure |
| **Return** | Pointer to the full structure containing the hash element |
| **Function** | Converts pointer to hash element into a pointer to the structure that HASH\_ELEM is embedded inside, by calculating its memory location. Used in main.c to access the full hash\_item from a hash\_elem. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bm, size\_t exSize) |
| **Parameter** | bm: Pointer of bitmap that you want to expand, exSize: #(bits to add) |
| **Return** | struct bitmap \*: Pointer of expanded bitmap if succeed, NULL if fail |
| **Function** | Increases the size of the bitmap by the specified number of bits and initializes new bits to 0 |