```c
#ifndef __GENQUEUE_H__
#define __GENQUEUE_H__

#include <stdlib.h>             /*for size_t*/

typedef enum{
        QUEUE_SUCCESS,
        QUEUE_UNINITIALIZED_ERROR,
        QUEUE_OVERFLOW_ERROR,
        QUEUE_DATA_NOT_FOUND_ERROR,
        QUEUE_DATA_UNINITIALIZED_ERROR
}QueueResult;

typedef struct Queue Queue;

typedef void (*DestroyItem)(void* _element);
typedef int (*ActionFunction)(const void* _element, void* _context);
/**
 * @brief Create a new queue with a size of _size
 * @param[in] _size - The size of the queue
 * @return Queue pointer - on success, NULL on failure
 *
 */
Queue* QueueCreate(size_t _size);

/**
 * @brief Unallocate a previously created queue
 * @param[in] _queue - the queue to unallocate
 * @param[in] _itemDestroy - Optional destrot function for item
 * @return void - no return value
 *
 */
void QueueDestroy(Queue** _queue, DestroyItem _itemDestroy);

/**
 * @brief Add a new item to the queue
 * @param[in] _queue - The queue to add the item to
 * @param[in] _item - the item to add to the queue
 * @return Queue_Result -
 * @retval QUEUE_SUCCESS - on seccess
 * @retval QUEUE_UNINITIALIZED_ERROR - if _queue is NULL
 * @retval QUEUE_DATA_UNINITIALIZED_ERROR - if _item is NULL
 * @retval QUEUE_OVERFLOW_ERROR - if there is no more room to add
 *
 */
QueueResult QueueInsert(Queue* _queue,void* _item);

/**
 * @brief Remove the first item in the queue
 * @param[in] _queue - the queue to remove the information from
 * @param[in] _item - A pointer to the information removed
 * @return Queue_Result -
 * @retval QUEUE_SUCCESS - on seccess
 * @retval QUEUE_UNINITIALIZED_ERROR - if _queue is NULL
 * @retval QUEUE_DATA_UNINITIALIZED_ERROR - if _item is NULL
 * @retval QUEUE_DATA_NOT_FOUND_ERROR - if queue is empty
 *
 */
QueueResult QueueRemove(Queue* _queue,void** _item);

/**
 * @brief Check if a given queue is empty
 * @param[in] _queue - The queue to check
 * @return size_t - return none zero if empty
 *
 */
size_t QueueIsEmpty(Queue* _queue);

/**
 * @brief ForEach function to implement on all elements in queue
 * @param[in] _queue - The queue
 * @param[in] _action - The action to implement on elements
 * @param[in] _context - context for action
 * @return size_t - the number times action was implemented
 *
 * @details if action returns 0 stop running and return the number of
times action run
 */
size_t QueueForEach(Queue* _queue, ActionFunction _action, void*
_context);

#endif          /*__GENQUEUE_H__*/
```