```
#ifndef __GENHEAP_H__
#define __GENHEAP_H__

#include <genvec.h>
/**
 * @brief Create a Binary heap generic data type over a generic vector
data type.
 * @author Author MuhammadZ (muhammadz@experis.co.il)
 * @see https://en.wikipedia.org/wiki/Binary_heap
 */


typedef enum Heap_ResultCode {
        HEAP_SUCCESS = 0,
        HEAP_NOT_INITIALIZED,
        HEAP_REALLOCATION_FAILED
} HeapResultCode;

typedef struct Heap Heap;


typedef int     (*ActionFunction)(const void *_elem, void * _context);
typedef int     (*LessThanComparator)(const void *_left, const void
*_right);

/**
 * @brief Dynamically create a new heap
 * @param[in] _vector - Vector that hold the elements, must be
initialized
 * @param[in] _pfLess - A less than comparator to be used to compare
elements
 * @return Heap * - on success
 * @retval NULL on fail
 *
 * @warning allocating and freeing the underlying vector is user
responsibility.
 */
Heap* HeapBuild(Vector* _vector, LessThanComparator _pfLess);

/**
 * @brief Deallocate a previously allocated heap
 * Frees the heap but not the underlying vector
 * @param[in] _heap - Heap to be deallocated.  On success will be
nulled.
 * @return Underlying vector
 */
Vector* HeapDestroy(Heap** _heap);

/**
 * @brief Add an element to the Heap preserving heap property.
 * @param[in] _heap - Heap to insert element to.
 * @param[in] _element - Element to insert.
 * @return success or error code
 * @retval HEAP_OK  on success
 * @retval HEAP_NOT_INITIALIZED  error, heap not initilized
 * @retval HEAP_REALLOCATION_FAILED error, heap could not reallocate
```

```
underlying vector
 */
HeapResultCode HeapInsert(Heap* _heap, void* _element);

/**
 * @brief Peek at element on top of heap without extracting it.
 * @param[in] _heap - Heap to peek to.
 * @return pointer to element, can be null if heap is empty.
 */
const void* HeapPeek(const Heap* _heap);

/**
 * @brief Extract element on top of heap and remove it.
 * @param[in] _heap - Heap to extract from.
 * @return pointer to element, can be null if heap is empty.
 */
void* HeapExtract(Heap* _heap);

/**
 * @brief Get the current size (number of elements) in the heap.
 * @param[in] _heap - Heap to extract from.
 * @return number of elements or zero if empty.
 */
size_t HeapSize(const Heap* _heap);

/**
 * @brief Iterate over all elements  in the heap from top to bottom.
 * @details The user provided ActionFunction _act will be called for
each element.
 * @param[in] _heap - Heap to iterate over.
 * @param[in] _act - User provided function pointer to be onvoked for
each element
 * @returns number of times the user functions was invoked
 */
size_t HeapForEach(const Heap* _heap, ActionFunction _act, void*
_context);


/**
 * @brief Sort a given vector of elments using a heap sort.
 * @param[in] _vector - vector to sort.
 * @param[in] _pfLess
 * @return number of elements or zero if empty.
 */
void HeapSort(Vector* _vec, LessThanComparator _pfLess);

#endif /*__GENHEAP_H__*/
```