#ifndef SORT\_H

#define SORT\_H

#include <stdio.h>

#include <stdlib.h>

/\* Comparison direction macros for bitonic sort \*/

#define UP 0

#define DOWN 1

/\*\*

\* enum bool - Enumeration of Boolean values.

\* @false: Equals 0.

\* @true: Equals 1.

\*/

typedef enum bool

{

false = 0,

true

} bool;

/\*\*

\* struct listint\_s - Doubly linked list node

\*

\* @n: Integer stored in the node

\* @prev: Pointer to the previous element of the list

\* @next: Pointer to the next element of the list

\*/

typedef struct listint\_s

{

const int n;

struct listint\_s \*prev;

struct listint\_s \*next;

} listint\_t;

/\* Printing helper functions \*/

void print\_array(const int \*array, size\_t size);

void print\_list(const listint\_t \*list);

/\* Sorting algoritms \*/

void bubble\_sort(int \*array, size\_t size);

void insertion\_sort\_list(listint\_t \*\*list);

void selection\_sort(int \*array, size\_t size);

void quick\_sort(int \*array, size\_t size);

void shell\_sort(int \*array, size\_t size);

void cocktail\_sort\_list(listint\_t \*\*list);

void counting\_sort(int \*array, size\_t size);

void merge\_sort(int \*array, size\_t size);

void heap\_sort(int \*array, size\_t size);

void radix\_sort(int \*array, size\_t size);

void bitonic\_sort(int \*array, size\_t size);

void quick\_sort\_hoare(int \*array, size\_t size);

#endif /\* \_GIDEON\_SORT\_H \*/