```c
#ifndef _LIST_H
#define _LIST_H

#define DATA_NOT_FOUND  -1
#define LIST_EMPTY      -2
#define SUCCESS         1
#define FAILURE         0
#define TRUE            1
#define FALSE           0

struct node;
typedef struct node node_t;
typedef node_t      list_t;
typedef int         data_t;
typedef int         len_t;
typedef int         flag_t;
typedef int         result_t;
typedef int         bool;

struct node
{
    data_t data;
    struct node *prev, *next;
};

/* List Interface Routines */

list_t          *create_list(void);
result_t        insert_beg(list_t *lst, data_t new_data);
result_t        insert_end(list_t *lst, data_t new_data);
result_t        insert_after_data(list_t *lst, data_t e_data, data_t new_data);
result_t        insert_before_data(list_t *lst, data_t e_data, data_t new_data);
result_t        delete_beg(list_t *lst);
result_t        delete_end(list_t *lst);
result_t        delete_data(list_t *lst, data_t d_data);

result_t        examine_beg(list_t *lst, data_t *p_data);
result_t        examine_end(list_t *lst, data_t *p_data);
result_t        examine_and_delete_beg(list_t *lst, data_t *p_data);
result_t        examine_and_delete_end(list_t *lst, data_t *p_data);

result_t        find(list_t *lst, data_t f_data);
void            display(list_t *lst);
bool            is_empty(list_t *lst);
len_t           len(list_t *lst);

data_t          *to_array(list_t *lst, len_t *p_len);
list_t          *to_list(data_t *p_arr, len_t len);
list_t          *merge(list_t *lst1, list_t *lst2);
list_t          *concat(list_t *lst1, list_t *lst2);

result_t        destroy_list(list_t **pp_lst);
```

```

/* List auxillary routines */
static void     g_insert(node_t *beg, node_t *mid, node_t *end);
static void     g_delete(node_t *node);
static node_t   *search_node(list_t *lst, data_t search_data);
static node_t   *get_node(data_t new_data);
/* Auxillary routines */
static void     *xcalloc(int nr_elements, int size_per_element);

#endif /* _LIST_H */


```