```c
#ifndef _LIST_H
#define _LIST_H

#include <stdio.h>

/* symbolic constants */
#define SUCCESS     1
#define TRUE        1
#define FALSE       0

#define LIST_DATA_NOT_FOUND     2
#define LIST_EMPTY              3

typedef int data_t;
typedef int status_t;
typedef int len_t;

typedef struct node node_t;
typedef node_t list_t;

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

/* interface functions */
list_t* create_list(void);

status_t insert_start(
            list_t* p_list,
            data_t new_data
        );
status_t insert_end(
            list_t* p_list,
            data_t new_data
        );
status_t insert_after(
            list_t* p_list,
            data_t e_data,
            data_t new_data
        );
status_t insert_before(
            list_t* p_list,
            data_t e_data,
            data_t new_data
        );

status_t get_start(
            list_t* p_list,
            data_t* p_start_data
        );
status_t get_end(
            list_t* p_list,
            data_t* p_end_data
        );
status_t pop_start(
            list_t* p_list,
            data_t* p_start
        );
status_t pop_end(
            list_t* p_list,
            data_t* p_end_data
        );

status_t remove_start(list_t* p_list);
status_t remove_end(list_t* p_list);
status_t remove_data(
            list_t* p_list,
            data_t r_data
        );
status_t clear_list(list_t* p_list);

status_t is_list_empty(list_t* p_list);
status_t find_in_list(
            list_t* p_list,
            data_t f_data
        );
len_t get_list_length(list_t* p_list);
```

```
81
82  void show_list(
83          list_t* p_list,
84          const char* msg
85      );
86
87
88  status_t destroy_list(list_t** pp_list);
89
90  /* concat immuable */
91  list_t* get_concated_list(
92              list_t* p_list_1,
93              list_t* p_list_2
94          );
95
96  /* concat mutable */
97  status_t concat_lists(
98              list_t* p_list_1,
99              list_t** pp_list_2
100         );
101
102 /* merge sorted lists */
103 list_t* merge_lists(
104             list_t* p_list_1,
105             list_t* p_list_2
106         );
107
108 /* reversal routines */
109 list_t* get_reversed_list(list_t* p_list);
110 status_t reverse_list(list_t* p_list);
111
112 /* list axuillary functions */
113
114 static void generic_insert(
115                 node_t* p_beg,
116                 node_t* p_mid,
117                 node_t* p_end
118             );
119 static void generic_delete(node_t* p_delete_node);
120 static node_t* search_node(
121                     list_t* p_list,
122                     data_t s_data
123                 );
124 static node_t* get_node(data_t new_data);
125
126 /* auxillary function */
127 static void* xcalloc(
128                 size_t nr_elements,
129                 size_t size_per_element
130             );
131
132 #endif /* _LIST_H */
133
```