```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// our node structure
struct node {
    int value;
    struct node *next;
};

// new function pointer type: list_add_f
typedef struct node * (*list_add_f)(struct node *head,
 struct node *n);

struct node *node_create(int value);
struct node *list_append(struct node *head, struct node *n);
struct node *list_prepend(struct node *head, struct node
 *n);
struct node *list_add_random(struct node *head, struct node
 *n);

void list_free(struct node *head);
void pfree(void *address);
void print_list(struct node *head);

// adds a node to the end of the linked list
// returns the new head of the linked list
struct node *list_append(struct node *head, struct node *n)
 {
    if (head == NULL)
        return n;

    struct node *cursor = head;
    while (cursor->next != NULL)
        cursor = cursor->next;

    cursor->next = n;
    return head;
}
```

```c
// adds a node to the end of the linked list
// returns the new head of the linked list
struct node *list_prepend(struct node *head, struct node
 *n) {
    if (head == NULL)
        return n;

    n->next = head;
    return n;
}

// don't care how it is added
// returns the new head of the linked list
struct node *list_add_random(struct node *head, struct node
 *n) {
    if (rand() % 2 == 0)
        return list_append(head, n);

    return list_prepend(head, n);
}

struct node *create_list( int count, int *values,
 list_add_f fn_ptr) {
    if (count <= 0 || values == NULL || fn_ptr == NULL)
        return NULL;

    struct node *head = node_create(values[0]);

    int i = 1;
    for ( ; i < count; ++i)
        head = fn_ptr( head, node_create(values[i]) );

    return head;
}

int main()
{
    //srand(0); // force fixed random behaviour
    srand(time(NULL)); // force random based on current time

    struct node *head;

    int vals[] = {1, 2, 3, 4, 5};
    head = create_list(5, vals, list_append);
    print_list(head);
    list_free(head);

    int vals2[] = {1, 2, 3, 4, 5};
    head = create_list(5, vals2, list_prepend);
    print_list(head);
    list_free(head);

    int vals3[] = {1, 2, 3, 4, 5};
    head = create_list(5, vals3, list_add_random);
    print_list(head);
    list_free(head);

    return 0;
}

// create a new node with value
struct node *node_create(int value) {
    struct node *n = (struct node*)malloc(sizeof(struct
     node));
    n->value = value;
    n->next = NULL;
    return n;
}

void print_list(struct node *head) {
    if (head == NULL) {
        printf("empty list!\n");
        return;
    }
    struct node *cursor = head;
    while (cursor != NULL) {
        printf("cursor element is : %d\n", cursor->value);
        cursor = cursor->next;
    }
}

void pfree(void *address) {
    struct node *tmp = (struct node*)address;
    printf("freeing %p %d\n", address, tmp->value);
    free(address);
}

void list_free(struct node *head)
{
    if (head == NULL)
```

```
        return;

    // at this point 1+ elements
    struct node *cursor = head;

    if (cursor->next == NULL) {
        // element 1 only case
        pfree(head);
        return;
    }

    // at this point 2+ cases
    struct node *tmp;
    while (cursor != NULL)  {
        tmp = cursor->next;
        pfree(cursor);
        cursor = tmp;
    }
}
```