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

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

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

// 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;
}

// 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;
}

int main() {
    struct node *head;

    head = node_create(28);
    head = list_append( head, node_create(14) );
    head = list_append( head, node_create(3) );
    head = list_append( head, node_create(7) );
```

```c
    print_list(head);

    list_free(head);

    return 0;
}

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;
        }
    }
```