```c
/***********************************************
 * LINKED_LIST.C: LEARN HOW TO USE A LINKED LIST *
 ***********************************************/

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

typedef struct _node {
        void* data;
        struct _node* next;
} node;

typedef struct _linked_list {
        node* head;
        node* tail;
        node* current;
} linked_list;

void initialize_list(linked_list *list){
        list->head = NULL;
        list->tail = NULL;
        list->current = NULL;
}

// Data specific function pointers
typedef int (*print_func)(void* data);
typedef void (*delete_func)(void* data);

// Create a new head node and add data to it.
void insert_first(linked_list* list, void* data){
        // Allocate memory for new node
        node* new = (node*) malloc(sizeof(node));
        assert(new != NULL);
        // Intitialize new node
        new -> data = data;
        new -> next = list->head;
        // Modify list pointers
        if (list->tail == NULL) {
                list->tail = new;
        }
        list->head = new;
}

// Delete the first node
void delete_first(linked_list* list){
        if(list->head != NULL){
                node* tmp = list->head;
                list->head = list->head->next;
                free(tmp);
        }
}

void print_list(linked_list* list, print_func print_data){
        printf("Linked list:\n");
        node* current = list->head;
        while(current != NULL){
                print_data(current->data);
                current = current->next;
        }
}

// Driver program
int main(){
        linked_list* list = malloc (sizeof(linked_list));
        assert(list != NULL);

        initialize_list(list);
        char *a= "This is a linked list of strings!";
```

```
70          char *b = "Nodes can be inserted and deleted dynamically.";
71          char *c = "Any type of data can be stored within the nodes.";
72          char *d = "This is the content of the tail node.";
73
74          insert_first(list,d);
75          insert_first(list,c);
76          insert_first(list,b);
77          insert_first(list,a);
78          print_list(list, (print_func) puts);
79
80          printf("\nLet's delete some nodes:\n");
81          delete_first(list);
82          delete_first(list);
83          delete_first(list);
84          print_list(list, (print_func) puts);
85
86          free(list);
87          return 0;
88  }
```