

CSc 352

# Linked List with Structs

Benjamin Dicken

# Linked List

- Previously build linked list without structs
- Node was defined as just: `typedef void* lln;`
- Implement using a struct instead:

```
typedef struct ListNode {  
    int value;  
    struct ListNode* next;  
} ListNode;
```

## Start simple:

```
typedef struct ListNode {  
    int value;  
    struct ListNode* next;  
} ListNode;
```

## Later, more advanced:

```
typedef struct ListNode {  
    void* value;  
    struct ListNode* prev;  
    struct ListNode* next;  
} ListNode;
```

```
typedef struct LinkedList {  
    int count;  
    ListNode root;  
} LinkedList;
```

```
typedef void* lln;
```

```
lln lln_create(int value) {  
    lln node = malloc(sizeof(int) + sizeof(lln));  
    if (node == NULL) {  
        fprintf(stderr, "ISSUE ALLOCATING NODE\n");  
        exit(1);  
    }  
    int* int_addr = ((int*)node);  
    lln* next_addr = (lln)(int_addr+1);  
    *int_addr = value;  
    *next_addr = NULL;  
    return node;  
}  
  
void lln_add(lln node, int value) {  
    int* int_addr = ((int*)node);  
    lln* next_addr = (lln)(int_addr+1);  
    if (*next_addr != NULL) {  
        lln_add(*next_addr, value);  
    } else {  
        *next_addr = lln_create(value);  
    }  
}
```

```
void lln_print(lln node) {  
    int* int_addr = ((int*)node);  
    lln* next_addr = (lln)(int_addr+1);  
    if (*next_addr != NULL) {  
        printf("[NODE (value=%d)] -> ", *int_addr);  
        lln_print(*next_addr);  
    } else {  
        printf("[NODE (value=%d)]\n", *int_addr);  
    }  
}  
  
int main() {  
    lln numbers;  
    numbers = lln_create(10);  
    lln_print(numbers);  
    lln_add(numbers, 20);  
    lln_add(numbers, 50);  
    lln_add(numbers, 30);  
    lln_print(numbers);  
    return 0;  
}
```

```
typedef void* lln;  
  
lln lln_create(int value) {  
    lln node = malloc(sizeof(int) + sizeof(lln));  
    if (node == NULL) {  
        fprintf(stderr, "ISSUE ALLOCATING NODE\n");  
        exit(1);  
    }  
    int* int_addr = ((int*)node);  
    lln* next_addr = (lln)(int_addr+1);  
    *int_addr = value;  
    *next_addr = NULL;  
    return node;  
}  
  
void lln_add(lln node, int value) {  
    int* int_addr = ((int*)node);  
    lln* next_addr = (lln)(int_addr+1);  
    if (*next_addr != NULL) {  
        lln_add(*next_addr, value);  
    } else {  
        *next_addr = lln_create(value);  
    }  
}
```

```
void lln_print(lln node) {  
    int* int_addr = ((int*)node);  
    lln* next_addr = (lln)(int_addr+1);  
    if (*next_addr != NULL) {  
        printf("[NODE (value=%d)] -> ", *int_addr);  
        lln_print(*next_addr);  
    } else {  
        printf("[NODE (value=%d)]\n", *int_addr);  
    }  
}  
  
int main() {  
    lln numbers;  
    numbers = lln_create(10);  
    lln_print(numbers);  
    lln_add(numbers, 20);  
    lln_add(numbers, 50);  
    lln_add(numbers, 30);  
    lln_print(numbers);  
    return 0;  
}
```

## Use the new struct instead

```
typedef struct ListNode {
    int value;
    struct ListNode* next;
} ListNode;

ListNode* ListNode_create(int value) {
    ListNode* node = malloc(sizeof(ListNode));
    if (node == NULL) {
        fprintf(stderr, "ISSUE ALLOCATING NODE\n");
        exit(1);
    }
    node->value = value;
    node->next = NULL;
    return node;
}

void ListNode_add(ListNode* node, int value) {
    if (node->next != NULL) {
        ListNode_add(node->next, value);
    } else {
        node->next = ListNode_create(value);
    }
}
```

```
void ListNode_print(ListNode* node) {
    if (node->next != NULL) {
        printf("[NODE (value=%d)] -> ", node->value);
        ListNode_print(node->next);
    } else {
        printf("[NODE (value=%d)]\n", node->value);
    }
}

int main() {
    ListNode* numbers;
    numbers = ListNode_create(10);
    ListNode_print(numbers);
    ListNode_add(numbers, 20);
    ListNode_add(numbers, 50);
    ListNode_add(numbers, 30);
    ListNode_print(numbers);
    return 0;
}
```

# Implement

```
bool ListNode_contains(ListNode* node, int value)
```

- Should return **true** if the linked list contains the **value** and **false** otherwise

```
typedef struct ListNode {  
    int value;  
    struct ListNode* next;  
} ListNode;
```

# Implement

```
void ListNode_sort(ListNode** node)
```

- Should sort the linked list passed in, treating the parameter as the root node of a linked list
- A **ListNode\*\*** is given, because the sorting process might need to change what the root node is!



# Modify the Implementation

Modify the implementation to:

- Have a char value pointer
- Use doubly-linked nodes
- Use a **LinkedList** struct

```
typedef struct ListNode {  
    char* value;  
    struct ListNode* prev;  
    struct ListNode* next;  
} ListNode;
```

```
typedef struct LinkedList {  
    int count;  
    ListNode* root;  
} LinkedList;
```