CSc 352

Basic Linked List

Benjamin Dicken

Implement a very simple LinkedList

- In this problem we should implement a very simple linked list
- A node will be represented by:

```
typedef void* lln;
```

- Will have very simple functionality:

```
lln lln_create(int value);
void lln_add(lln node, int value);
void lln_print(lln node);
```

- Then, test it in main!

```c
typedef void* lln;

lln lln_create(int value) {
  ?
}

void lln_add(lln node, int value) {
  ?
}

void lln_print(lln node) {
  ?
}

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;
}
```

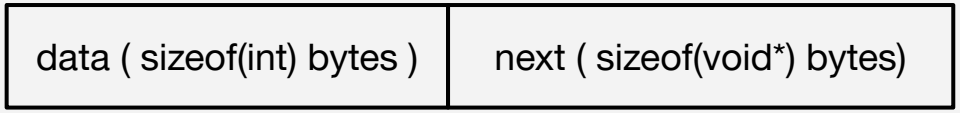
```c
typedef void* lln;

lln lln_create(int value) {
  ?
}

void lln_add(lln node, int value) {
  ?
}

void lln_print(lln node) {
  ?
}

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;
}
```



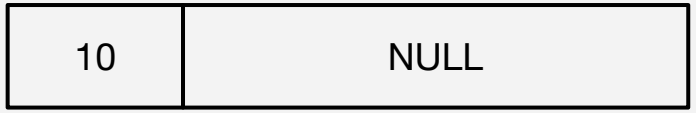
```c
typedef void* lln;

lln lln_create(int value) {
  ?
}

void lln_add(lln node, int value) {
  ?
}

void lln_print(lln node) {
  ?
}

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;
}
```



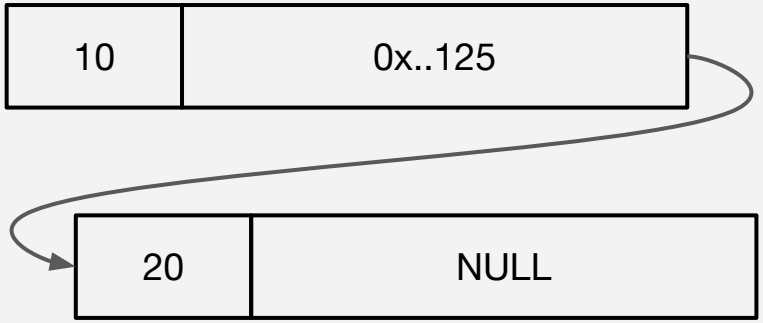
```c
typedef void* lln;

lln lln_create(int value) {
  ?
}

void lln_add(lln node, int value) {
  ?
}

void lln_print(lln node) {
  ?
}

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;
}
```



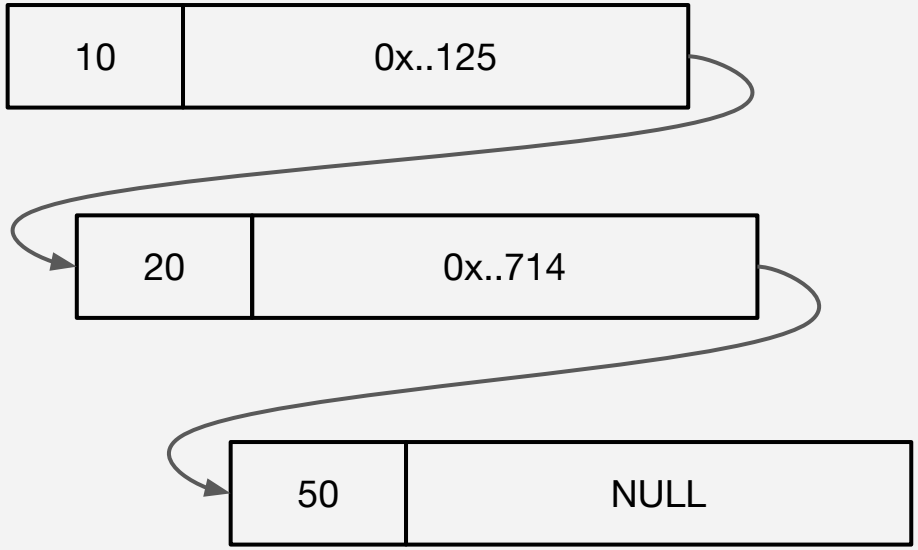
```c
typedef void* lln;

lln lln_create(int value) {
  ?
}

void lln_add(lln node, int value) {
  ?
}

void lln_print(lln node) {
  ?
}

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;
}
```



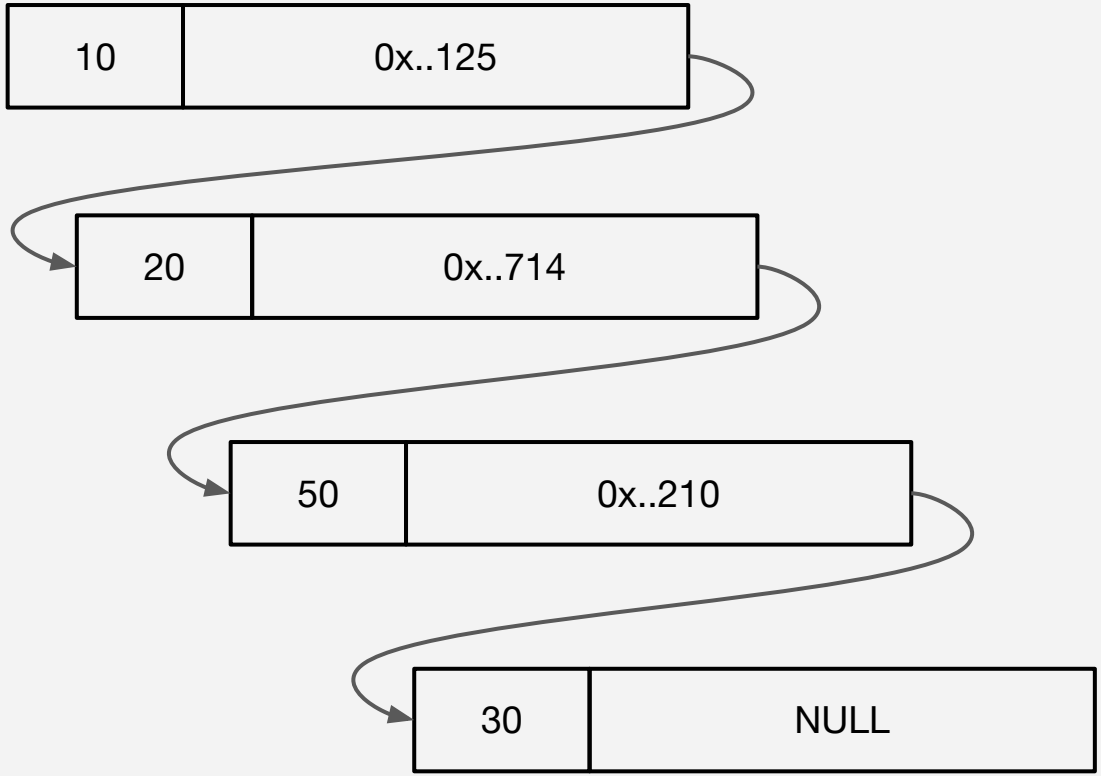
```c
typedef void* lln;

lln lln_create(int value) {
  ?
}

void lln_add(lln node, int value) {
  ?
}

void lln_print(lln node) {
  ?
}

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;
}
```



Implement a very simple LinkedList

- In this problem we should implement a very simple linked list
- A node will be represented by:

```
typedef void* lln;
```

- Will have very simple functionality:

```
lln lln_create(int value);
void lln_add(lln node, int value);
void lln_print(lln node);
```

- Then, test it in main!

```c
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);
  }
}
```

```c
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;
}
```

Activity

```c
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);
  }
}
```

```c
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;
}
```

Implement **`lln_free`**