CS 161

Introduction to CS I

Lecture 26

- Deleting recursive data structures
- More recursion power

Final Week 9 tips

- Check Canvas for any missing grades
  - Notify cs161-020-ta@engr.orst.edu by next Wednesday (3/11)
  - **Except:** Missing peer grades for Assign. 2 and 3 were recently set to 0. Normally these points are given when you demo. If you missed a demo, you may incorrectly have a 0 (never graded). These are now being re-graded, so don't send an email about these unless they are still 0 next Monday.
  - Final grades are rounded (89.4 -> 89; 89.5 -> 90)
- Assignment 6 will be worth 80 points
  - Worth doing if any previous assignment earned < 80 points
  - Worth doing if you want practice with recursion ☺

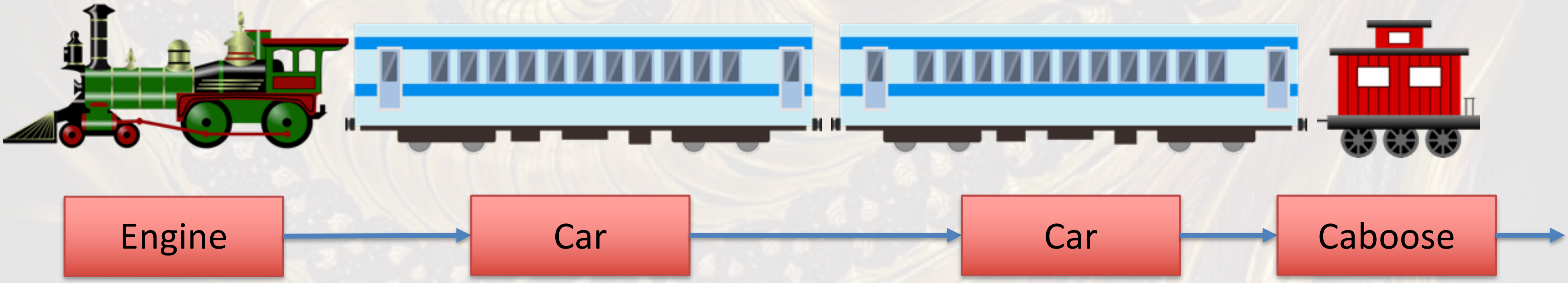
Proficiency demo in week 10

- Go to your registered lab (or contact TAs)
- To prepare:
  - Review 1D arrays, 2D arrays, and C-style strings
  - Practice: Give yourself 50 minutes to try one or more of the sample prompts
  - Design on paper before you start coding
  - Take a deep breath!
- Any questions about what to expect?

Review: Recursive data structures

- Let's model a train
  - Train = one or more train_car items, ending with a caboose

```
struct train_car {
   string kind;
   train_car* next_car;
};
```



Engine

Car

Car

Caboose

Deleting recursive data structures

- Create the train:

```
train_car* my_train = new train_car;
my_train->kind = "Engine";
my_train->next_car = NULL;

int n_cars = rand()%10 + 1;
add_cars(my_train, n_cars);
```

Deleting recursive data structures

- Delete a train:

```
train_car* my_train = new train_car;
my_train->kind = "Engine";
my_train->next_car = NULL;

int n_cars = rand()%10 + 1;
add_cars(my_train, n_cars);

delete my_train;
```

This deletes the first train_car (Engine) only. The rest are lost forever.

Deleting recursive data structures

- Instead, let's delete the train with a recursive function:

```
train_car* my_train = new train_car;
my_train->kind = "Engine";
my_train->next_car = NULL;

int n_cars = rand()%10 + 1;
add_cars(my_train, n_cars);

delete_train(my_train);
my_train = NULL;
```

See lec26-recur-train.cpp

Deleting recursive data structures

```
struct train_car {
  string kind;
  train_car* next_car;
};
```

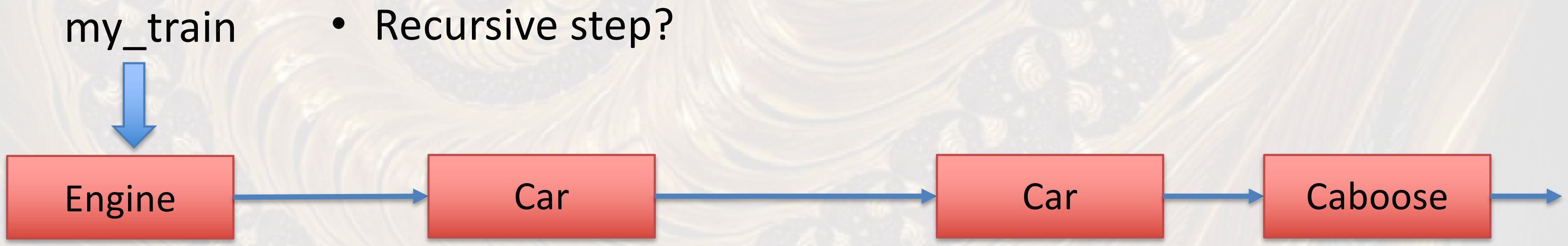
- How did we create the train?

```
void add_cars(train_car* t, int n_cars) {
  t->next_car = new train_car;
  t->next_car->next_car = NULL;
  if (n_cars == 1) {
    t->next_car->kind = "Caboose";
  } else {
    t->next_car->kind = "_***_";
    add_cars(t->next_car, n_cars-1);
  }
}
```

Deleting recursive data structures

```
struct train_car {
   string kind;
   train_car* next_car;
};
```

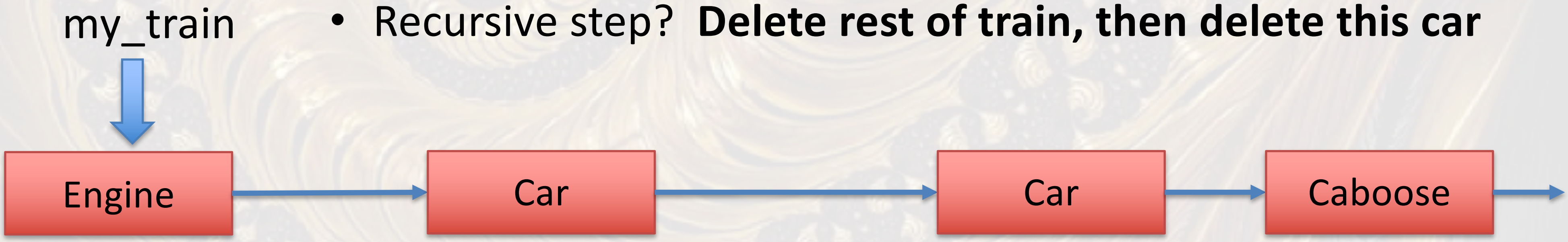
- Delete a train:
  - Wait to delete the current train_car until the rest of the train is gone
    - Base case?
    - Recursive step?



Deleting recursive data structures

```
struct train_car {
   string kind;
   train_car* next_car;
};
```

- Delete a train:
  - Wait to delete the current train_car until the rest of the train is gone
    - Base case? **Caboose**
    - Recursive step? **Delete rest of train, then delete this car**



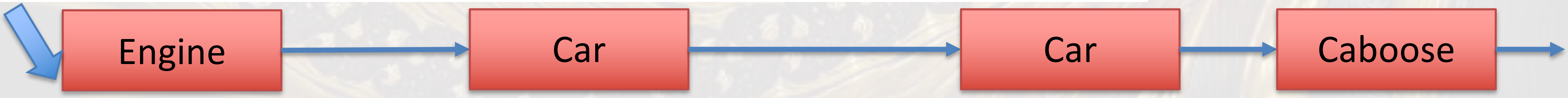
See lec26-recur-train.cpp

Your turn: Delete a train

```
struct train_car {
   string kind;
   train_car* next_car;
};
```

- Delete a train:

```
void delete_train(train_car* t) {
   if (t->kind == "Caboose")   /* base case */
      delete t;
   else {    /* recursive call */
      /* Delete the rest of the train first */
      delete_train(t->next_car);
      /* Now delete this car */
      delete t;
   }
}
```



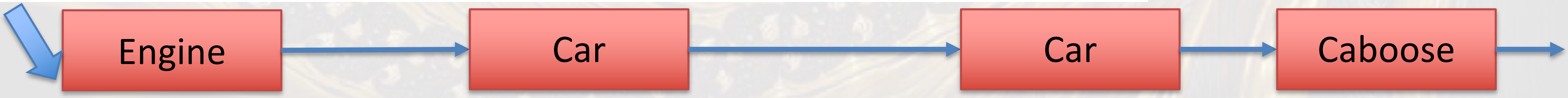
How NOT to delete a train

```
struct train_car {
  string kind;
  train_car* next_car;
};
```

- Delete a train:

```
void delete_train(train_car* t) {
  if (t->kind == "Caboose")  /* base case */
    delete t;
  else {   /* recursive call */
    /* Delete this car */
    delete t;
    /* Delete the                  train */
    delete_train
  }
}
```

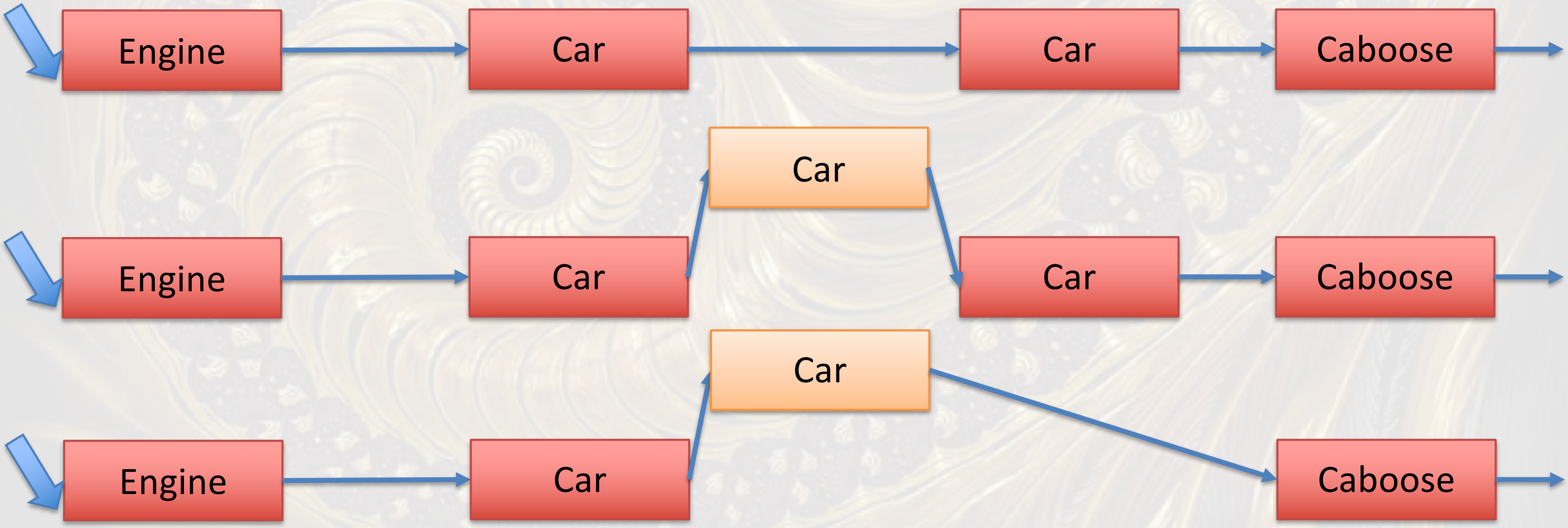
Seg fault



Oregon State University
College of Engineering

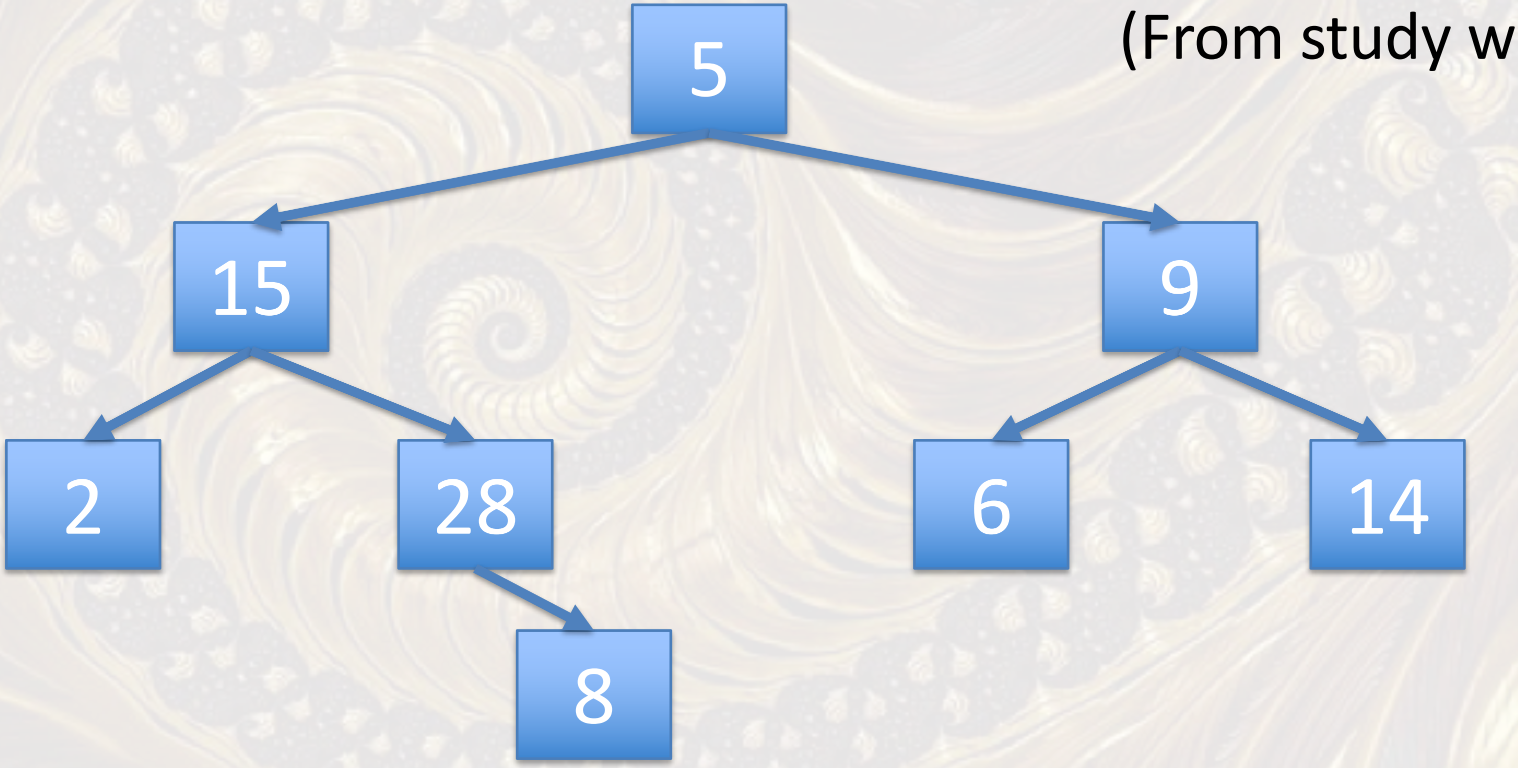
Our train_car is a linked list

- Add or remove cars as needed by reassigning pointers



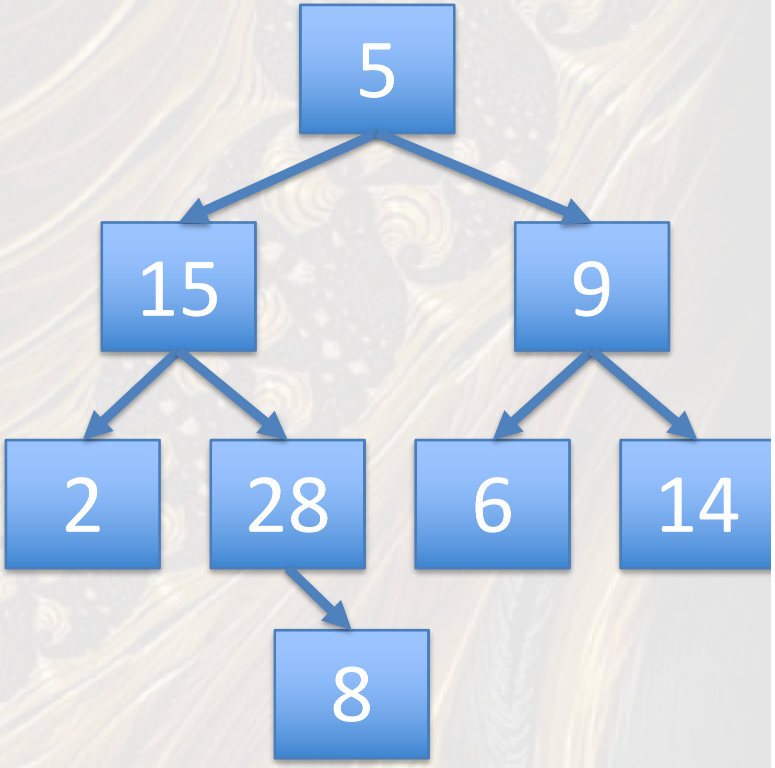
What if each struct has two pointers? (Tree)

(From study worksheet 9)



Your turn: Define a box struct

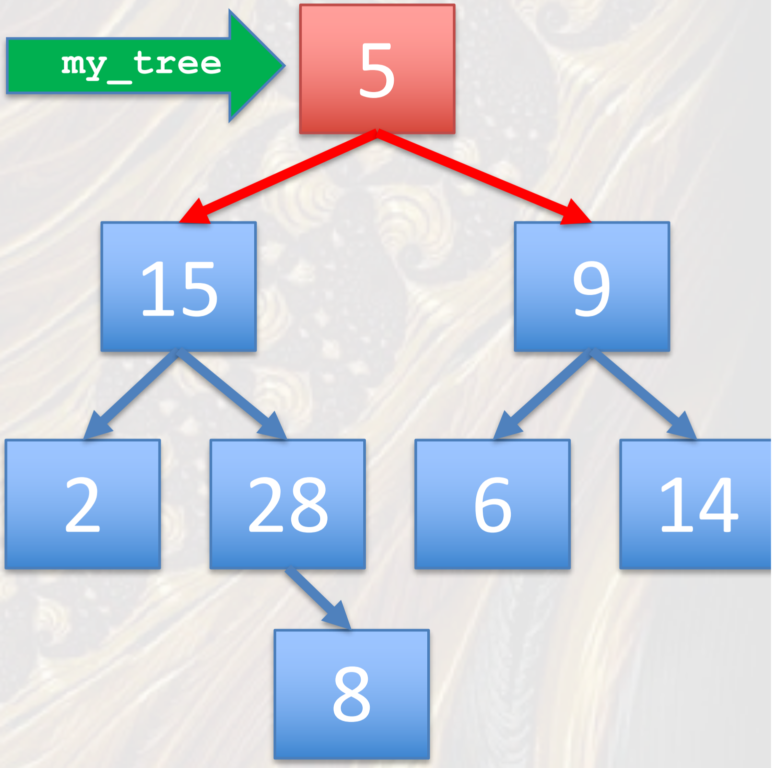
```
struct box {
  int value;
  box* left;
  box* right;
};
```



Your turn: Set up level 1

```
struct box {
  int value;
  box* left;
  box* right;
};
```

```
box* my_tree = new box;
my_tree->value = 5;
my_tree->left = NULL;
my_tree->right = NULL;
```

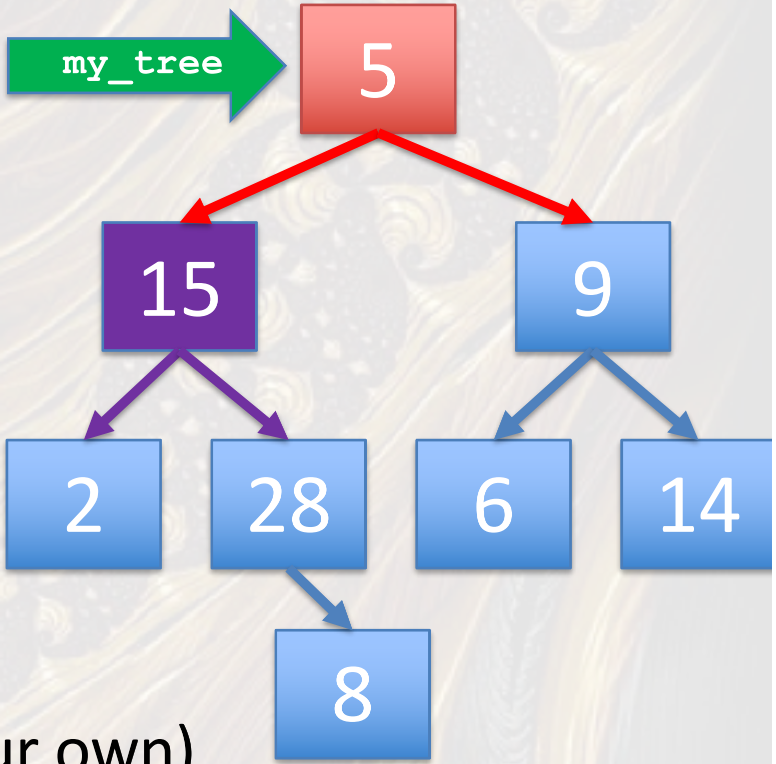


See lec26-recur-tree.cpp

Your turn: Set up level 2 (left child)

```
struct box {
  int value;
  box* left;
  box* right;
};
```

```
my_tree->left = new box;
my_tree->left->value = 15;
my_tree->left->left = NULL;
my_tree->left->right = NULL;
```

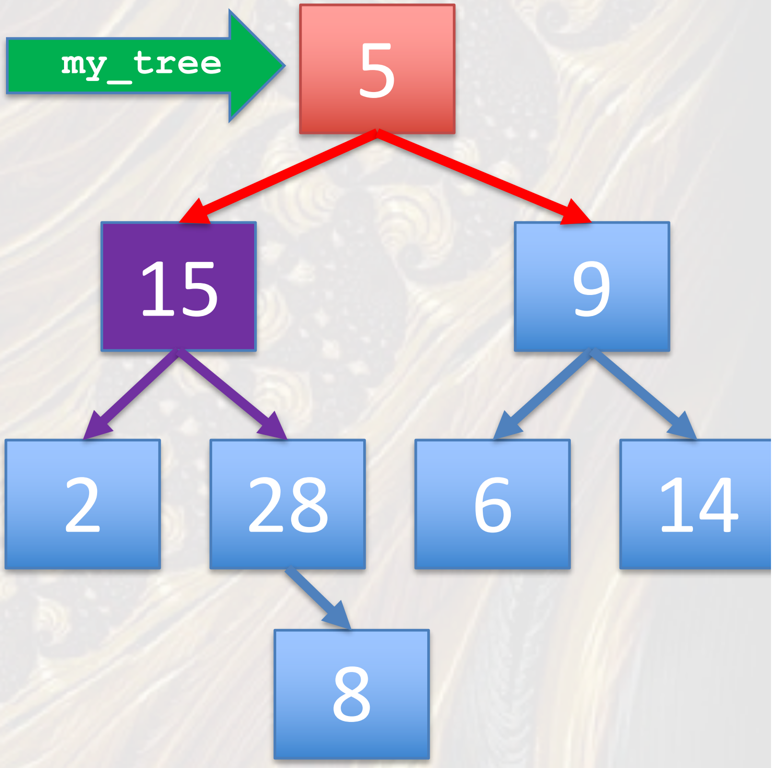


Same process for right child (try it on your own)

See lec26-recur-tree.cpp

Your turn: Delete the tree

```
void delete_tree(box* b) {
  if (b == NULL) /* base case */
    return;
  else {
    /* delete sub-trees first */
    delete_tree(b->left);
    delete_tree(b->right);
    /* now delete this box */
    delete b;
   }
}
```

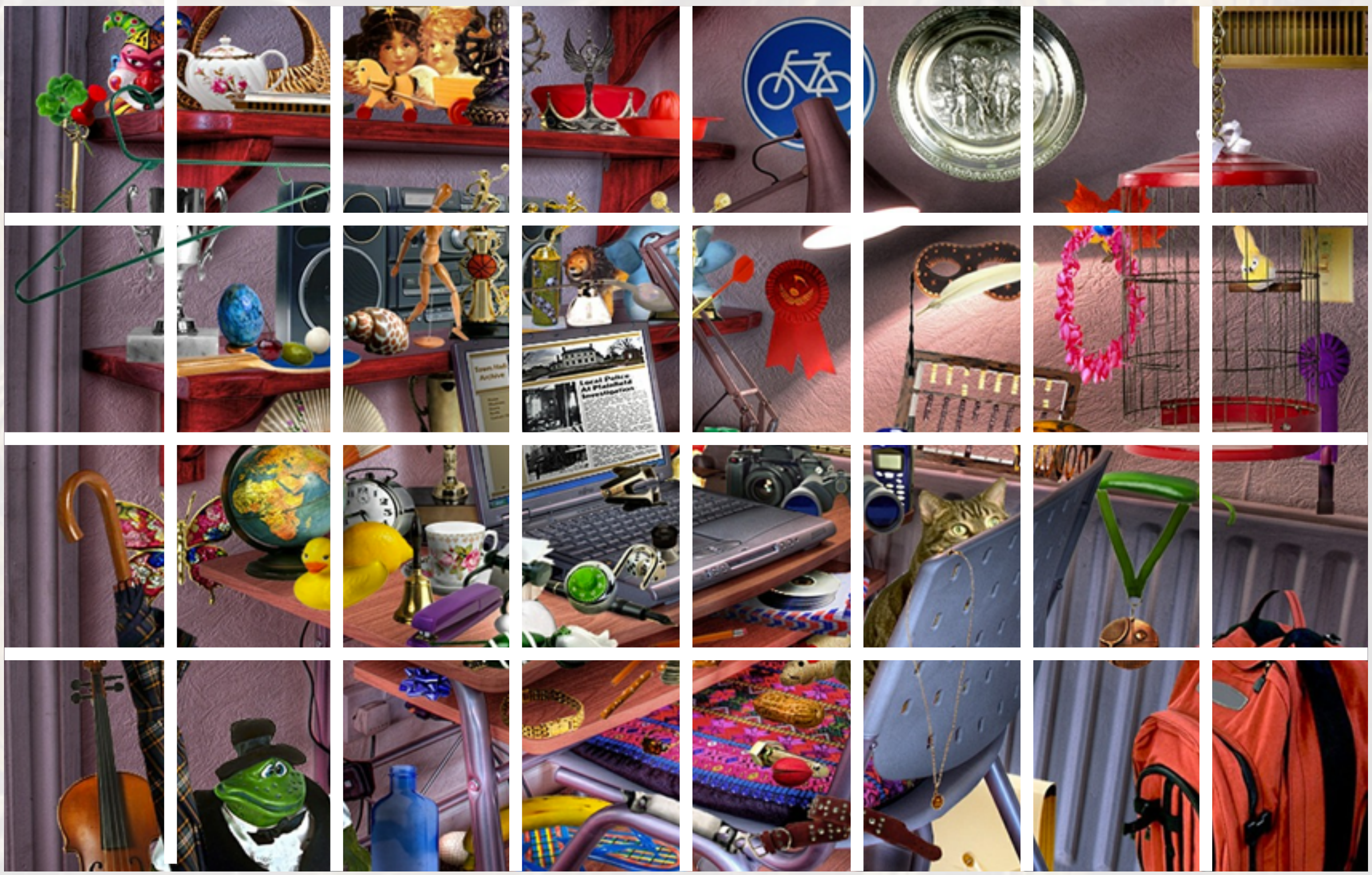


Recursion simplifies tasks: searching

- Where is the combination lock?

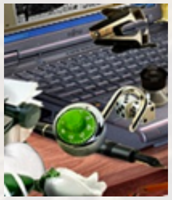
Recursion simplifies tasks: searching

- Where is the combination lock?



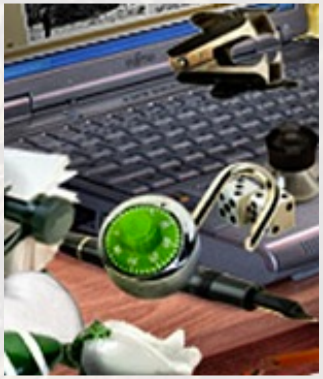
Recursion simplifies tasks: searching

- Where is the combination lock?



Recursion simplifies tasks: searching

- Where is the combination lock?



- Recursive definition of search_lock(image):
  - **Base case:** search_lock(small image) = look at image
  - **Recursive step:** search_lock(big image) = search_lock(half1) or search_lock(half2)

What ideas and skills did we learn today?

- How to delete recursive data structures
  - With a recursive function
- Data structure with single pointer: linked list
- Data structure with two pointers: tree
- How recursion can help break down bigger problems

Week 9 nearly done!

❑ Attend lab (laptop required)

❑ Read Rao lesson 7 (pp. 158-161)

Read Miller lecture 8:
http://www.doc.ic.ac.uk/~wjk/C++Intro/RobMillerL8.html

❑ **Assignment 5** (due **Sunday, March 8**)

See you Monday!