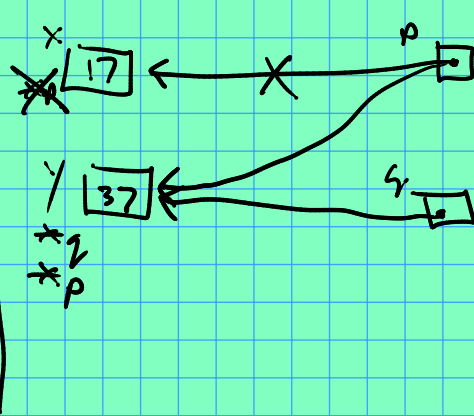


```

int x;
x = 17;
int y;
y = 37;
int * p;
p = &x;
int * q = &y;
p = q;

```



```

&p (type int**)
&q (type int**)

```

```

int * A = new int[5000000];
A = 0;

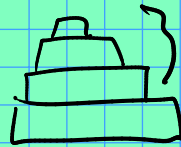
```

// if you're paranoid, maybe look
// at valgrind or other profiling tools

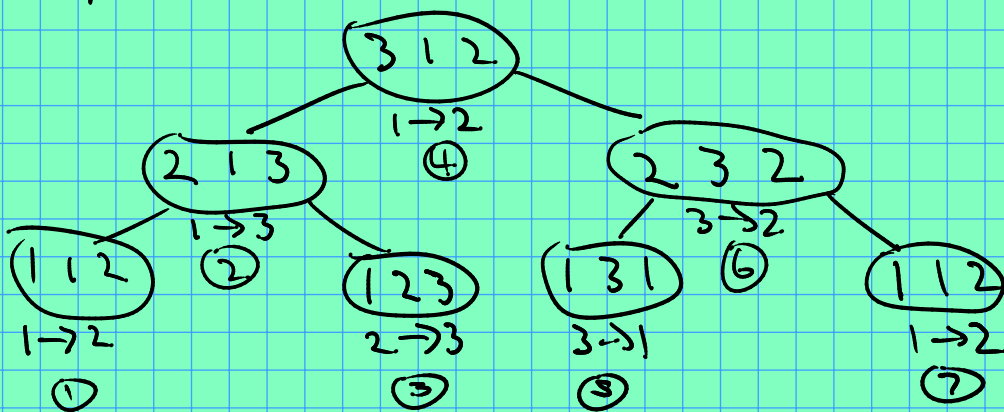
```

towers(n, s, e) {
    if (n == 0) return;
    towers(n-1, s, 6-s-e);
    cout << "move 1 from " << s << " to " << e << "\n";
    towers(n-1, 6-s-e, e);
}

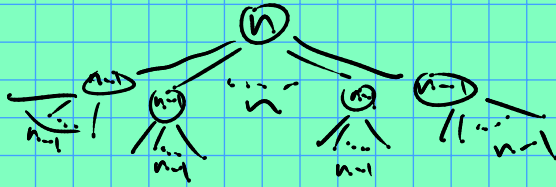
```



Example trace:



Computing permutations:



Why do we need copy const. & assignment (& destructor)
for classes w/ dynamic memory?

destructor — should be obvious — free dynamically
allocated memory once object goes out
of scope.

copy const. & assignment: make sure there is a
1-1 correspondence between dynamic allocations
& objects (so no two objects share memory).

Note! default behavior (member-wise assignment)
does not have this property

