// 368 Worksheet: Constructors+Destructors

```
class A {
  int x;
public:
  A(int x) : x(x) {
    cout << "Create A\n";
  }
  ~A() {
    cout << "Destroy A\n";
  }
};
class B {
  int y;
public:
  B(int y) : y(y) { cout << "Create B\n"; }
  ~B() { cout << "Destroy B\n"; }
};
class C {
  A a; B* b;
public:
  C(int x, int y) : b(new B(y)), a(A(x)) {
    cout << "Create C\n";
  }
  ~C() {
    cout << "Destroy C\n";

  }
};
```

addr	memory	notes
20		
24		
28		
32		val, val.a
36		val.b
40		
…		
100		*val.b

```
void test() {
  auto val = C(3,4);
}
```

1. fill in memory layout table, just prior to return of **`test`** function; label heap/frames
2. put a ✔️ by corresponding to each constructor call next to the addr of **`this`**
3. what is printed when **`val`** is initialized?
4. fill in changes to the memory table if "`auto val2 = val;`" was added to **`test`**
5. when **`test`** returns, what happens? Put an ✗ for each destructor invocation next to the associated address ("addr" column) and cross out any released memory ("memory" column)
6. what memory leaked?
7. what could we add to ~C() to avoid leaking?
8. what other functions should we worry about besides the destructor?