333 Section 6 SOLUTIONS - C++ Casting and Inheritance

Exercise 1

For each of the following, write down if we are using static dispatch, dynamic dispatch, or triggering a compile error.

```
class Base {
  void Foo();             static dispatch
  void Bar();             static dispatch
  virtual void Baz();     dynamic dispatch
};

class Derived : public Base {
  virtual void Foo();     dynamic dispatch (For more derived class)
  void Bar() override;    compiler error!
  void Baz();             dynamic dispatch
};
```

If there are any style mistakes or bugs, how would you fix them?

```
1.
class Derived : public Base {
  ...
  void Bar() override; → void Bar();
  ...
};

Using static dispatch, fixes compiler error.

2.
class Derived : public Base {
  ...
  void Baz(); → void Baz() override;
};

Will still be dynamic dispatch, but override with Derived's
implementation.
```

Exercise 2

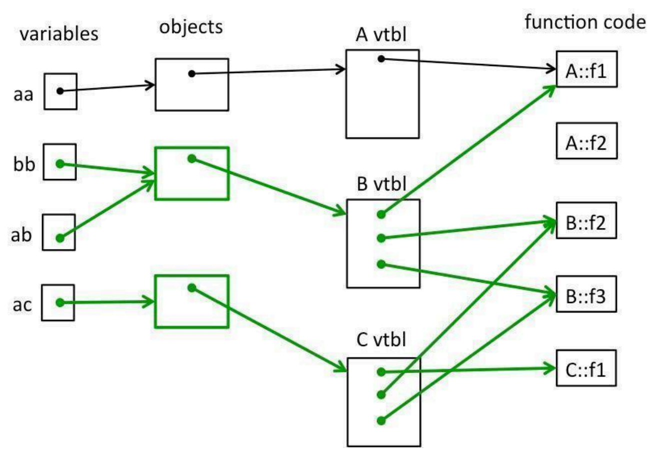
Consider the program on the following page, which does compile and execute with no errors, except that it leaks memory (which doesn't matter for this question).

(a) Complete the diagram on the next page by adding the remaining objects and all of the additional pointers needed to link variables, objects, virtual function tables, and function bodies. Be sure that the order of pointers in the virtual function tables is clear (i.e., which one is first, then next, etc.). One of the objects and a couple of the pointers are already included to help you get started.

(b) Write the output produced when this program is executed. If the output doesn't fit in one column in the space provided, write multiple vertical columns showing the output going from top to bottom, then successive columns to the right

```
#include <iostream>
using namespace std;

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl; }
  void f2() { cout << "A::f2" << endl; }
};
class B : public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl; }
  virtual void f2() { cout << "B::f2" << endl; }
};
class C : public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```



```
int main() {
  A* aa = new A();
  B* bb = new B();
  A* ab = bb;
  A* ac = new C();
  aa->f1();
  cout << "---" << endl;
  bb->f1();
  cout << "---" << endl;
  bb->f2();
  cout << "---" << endl;
  ab->f2();
  cout << "---" << endl;
  bb->f3();
  cout << "---" << endl;
  ac->f1();
  return EXIT_SUCCESS;
}
```

```
Output:
A::f2
A::f1
---
A::f2
A::f1
---
B::f2
---
A::f2
---
A::f2
A::f1
B::f3
---
B::f2
```

```
C::f1
```