Inheritance and Dynamic Binding

Consider the following two classes[1]:

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

class A {
  protected:
    int i;

  public:
    A(int i) : i(i) {}

    int f() const { return 2 * i; }

    int g() const { return 3 * i; }

    virtual int h() const { return 4 * i; }

    friend ostream& operator<<(ostream& o, const A& a) {
      return o << "A( i: " << a.i << " )";
    }
};

class B : public A {
  int j;

  public:
    B(int i, int j) : A(i), j(j) { }

    int g() const { return i + j; }

    virtual int h() const { return 40 * i; }

    friend ostream& operator<<(ostream& o, const B& b) {
      return o << "B( i: " << b.i << ", j: " << b.j << " )";
    }
};
```

[1]The sole aim of this exercise is to test the understanding of inheritance and dynamic binding. To this end the names of classes and their members where chosen to be brief and not to convey any meaning. This is not an example of good programming style.

What is the output of the following program?

```
int main() {
  A a1(3), a2(5);
  cout << " a1 is " << a1 << "; a2 is " << a2 << endl;

  a1 = a2;
  cout << "after a1 = a2:" << endl
    << " a1 is " << a1 << endl
    << " a1.f() == " << a1.f() << endl
    << " a1.g() == " << a1.g() << endl
    << " a1.h() == " << a1.h() << endl;

  B b(7, 2);
  cout << " b is " << b << endl
    << " b.f() == " << b.f() << endl
    << " b.g() == " << b.g() << endl
    << " b.h() == " << b.h() << endl;

  a1 = b;
  cout << "after a1 = b:" << endl
    << " a1 is " << a1 << endl
    << " a1.f() == " << a1.f() << endl
    << " a1.g() == " << a1.g() << endl
    << " a1.h() == " << a1.h() << endl;

  A* pa = new A(5);
  B* pb = new B(7, 2);
  cout << " *pa is " << *pa << endl
    << " pa->f() == " << pa->f() << endl
    << " pa->g() == " << pa->g() << endl
    << " pa->h() == " << pa->h() << endl;
  cout << " *pb is " << *pb << endl
    << " pb->f() == " << pb->f() << endl
    << " pb->g() == " << pb->g() << endl
    << " pb->h() == " << pb->h() << endl;

  pa = pb;
  cout << "after pa = pb:" << endl
    << " *pa is " << *pa << endl
    << " *pb is " << *pb << endl
    << " pa->f() == " << pa->f() << endl
    << " pa->g() == " << pa->g() << endl
    << " pa->h() == " << pa->h() << endl;}
```