Inheritance and Dynamic Binding

Consider the following two classes¹: 1 #include <iostream> using namespace std; 3 4 class A { 5 protected: 6 int i; 7 8 public: $A(int i) : i(i) \{\}$ 9 10 int f() const { return 2 * i; } 11 12 int g() const { return 3 * i; } 13 14 virtual int h() const { return 4 * i; } 15 16 Assignment Project Exam Help 17 18 19 20 }; 21 class B : puhttps://powcoder.com 22 23 int j; 24 ublic: Add WeChat powcoder public: 25 26 27 int g() const { return i + j; } 28 29 virtual int h() const { return 40 * i; } 30 31 friend ostream& operator<<(ostream& o, const B& b) { 32 return o << "B(i: " << b.i << ", j: " << b.j << ")"; 33 } 34 35 };

¹The 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()  {
1
 2
     A a1(3), a2(5);
     cout << " a1 is " << a1 << "; a2 is " << a2 << endl;
3
 4
5
     a1 = a2;
      cout << "after a1 = a2:" << endl
6
7
       << " a1 is " << a1 << endl
       << " a1.f() == " << a1.f() << endl
8
       << " a1.g() == " << a1.g() << endl
9
       << " a1.h() == " << a1.h() << endl;
10
11
12
     B b(7, 2);
13
      cout << " b is " << b << endl
        << " b.f() == " << b.f() << endl
14
15
       << " b.g() == " << b.g() << endl
       << " b.h() == " << b.h() <math><< endl;
16
17
18
     a1 = b;
     cou Assignment Project Exam Help
19
20
       << " a1.f() == " << a1.f() << endl</pre>
21
       << " a1.g() == " << a1.g() << endl
22
             al https://powcoder.com
23
24
25
     A* pa = new A(5);
     \begin{array}{ll} {\rm B*\ pb = new\ B(7,\ 2);}\\ {\rm cout\ <<\ "\ pa->f() ==\ "} \end{array} \\ \begin{array}{ll} {\rm WeChat}\\ {\rm powcoder} \end{array}
26
27
28
       << " pa->g() == " << pa->g() << endl
29
       << " pa->h() == " << pa->h() << endl;
30
      \operatorname{cout} << " *pb is " << *pb << endl
31
       << " pb->f() == " << pb->f() <math><< endl
32
       << " pb->g() == " << pb->g() << endl
33
34
       << " pb->h() == " << pb->h() << endl;
35
36
     pa = pb;
37
      cout << "after pa = pb:" << endl
       << " *pa is " << *pa << endl
38
       << " *pb is " << *pb << endl
39
       << " pa->f() == " << pa->f() << endl
40
       << " pa->g() == " << pa->g() << endl
41
       << " pa->h() == " << pa->h() << endl;}
42
```