PROGRAMACIÓN ORIENTADA A OBJETOS

EXAMEN 2

1. Dado el siguiente programa:

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  class B{
6.      public:
7.
8.          void f(){cout << "f() de B"<< endl;}
9.          virtual void g() { cout << "g() de B" << endl;}
10.         virtual void h()=0;
11.         virtual ~B(){};
12.     protected:
13.         int b;
14. };
15.
16. class D1: virtual public B{
17.     public:
18.         void f(){cout << "f() de D1"<< endl;}
19.         virtual void g(){cout << "g() de D1" << endl;}
20.         void h(){cout << "h() de D1"<<endl;}// Se debe de redifinir el virtual puro
21.     protected:
22.         int d1;
23. };
24.
25. class D2: virtual public B{
26.   public:
27.     void f(int i){cout << "f(" << i <<") de D2" << endl;}
28.     virtual void h(){ cout << "h() de D2 "<< endl;}
29.
30.    protected:
31.    int d2;
32. };
33.
34. class D3: public D1{
35.     public:
36.         void g(){cout << "g() de D3" << endl;}
37.         void h(){ cout << "h() de D3" << endl;}
38.     private:
39.         int d3;
40. };
41.
```

```
42. class D4: public D1,public D2{
43.     public:
44.         void h(){cout << "h() de D4" << endl;}
45.     private:
46.         int d4;
47. };
48.
49. void f(B& b){
50.     cout <<"f() externa" << endl;
51.     b.f();
52.     b.g();
53.     b.h();
54. }
55.
56. int main(){
57.     B b;
58.     B *pB;
59.     D1 d1;
60.     D2 d2;
61.     D3 d3;
62.     D4 d4;
63.     f(d1);
64.     f(d2);
65.     f(d3);
66.     f(d4);
67.     d4.D1::f();
68.     d4.D2::f(5);
69.     d4.D2::f(3.7);
70.     d4.g();
71.     d4.h();
72.     pB=new D4;
73.     pB->f();
74.     pB->D4::f(3);
75.     pB->g();
76.     pB->h();
77.     delete pB;
78. }
```

a) Corrija los errores, si los hay, en la definición de las clases B, D1, D2, D3 y D4. Para cada clase enumere sus miembros declarados o definidos explícitamente

b) Diga si el programa provoca algún error de compilación o de ejecución y porqué. Si lo tiene, modifique el código adecuadamente para solucionarlo.

2. Dada la clase paramétrica:

```
template<class T1, class T2> class par{
public:
    par(): prime(T1()), segun(T2()){}
    par(const T1& x, const T2& y): prime(x), segun(y){}
    T1 primero() const {return prime;}
    T1& primero(){return prime;}
    T2 segundo() const {return segun;}
    T2& segundo(){return segun;}
private:
    T1 prime;
    T2 segun;
};
```

a) Sobrecargar operador << para la plantilla de clase par

b) Defina un tipo racional a partir de ella para representar número racionales. Sobrecargue el operador suma de nº racionales

c) Defina una clase complejo con parte real e imaginaria de tipo double. Sobrecargue el operador de autosuma (+=) de nº complejos