/* Scrivere un template di classe SmartP <T> di puntatori smart a T che definisca assegnazione profonda, costruzione di copia profonda, e distruzione profonda. Il template SmartP<T> dovrà essere dotato di interfaccia pubblica che permetta di compilare correttamente il seguente codice, la cui esecuzione dovrà provocare esattamente le stampe riportate nei commenti. */

```
Class C {
Int * p;
C() : p (new int (5){}
};

int main (){
consti nt a = 1; const int * p = &a;
SmartP<int> r;
SmartP<int> s(&a);
SmartP<int> t(s);
cout<< *s << " " << *t << *p ;   // 1 1 1
*s = 2; *t =3;
cout<< *s  << *t << *p ;   // 2 3 1
r = t; *r = 4;
cout<< *r << *s << *t << *p ;   // 4 2 3 1
cout<< (s==t) << (s != p)  ;   // 0 1
C c; SmartP<c> x(&c);
cout << *(c.p) << *(x->p);  // 5 5
*(c.p) = 6;
 cout<< *s(c.p)<< *(x->p) ;   // 6 6
SmartP<C> *q = new SmartP<c>(&c);
delete q;
cout <<*(x->p) // 6s


}
```