Accessing Inherited Functions

You are given three classes *A, B* and *C*. All three classes implement their own version of *func*.

In class *A*, *func* multiplies the value passed as a parameter by 2:

```
class A
{
   protected:
      void func(int  & a)
      {
         a=a*2;
      }
 }
```

In class *B*, *func* multiplies the value passed as a parameter by 3:

```
class B
{
   protected:
      void func(int  & a)
      {
         a=a*3;
      }
 }
```

In class *C*, *func* multiplies the value passed as a parameter by 5:

```
class C
{
   protected:
      void func(int  & a)
      {
         a=a*5;
      }
 }
```

You are given a class *D*:

```
class D
{

   int val;
   public:
      //Initially, val is 1
       D()
       {
         val=1;
       }


       //Implement this function
       void update_val(int new_val)
       {


       }

}
```

You need to modify the class *D* and implement the function `update_val` which sets *D*'s *val* to *new_val* by

manipulating the value by only calling the *func* defined in classes *A, B* and *C*.

It is guaranteed that *new_val* has only $2, 3$ and 5 as its prime factors.

Input Format

Implement class *D*'s function *update_val*. This function should update *D*'s *val* only by calling *A, B* and *C*'s *func*.

Constraints

$1 \leq new\_val \leq 10000$
Note: The *new_val* only has $2, 3$ and 5 as its prime factors.

Sample Input

$new\_val = 30$

Sample Output

A's *func* will be called once.
B's *func* will be called once.
C's *func* will be called once.

Explanation

Initially, $val = 1$.

A's *func* is called once:

```
val = val*2
val = 2
```

B's *func* is called once:

```
val = val*3
val = 6
```

C's *func* is called once:

```
val = val*5
val = 30
```