Example 72. Solve $x \equiv 4 \pmod 5$, $x \equiv 10 \pmod{13}$.

Solution. $x \equiv 4 \cdot 13 \cdot \underbrace{13^{-1}_{\bmod 5}}_{2} + 10 \cdot 5 \cdot \underbrace{5^{-1}_{\bmod 13}}_{-5} \equiv 104 - 250 \equiv 49 \pmod{65}$

Check. Since it is easy to do so, we should quickly check our answer: $49 \equiv 4 \pmod 5$, $49 \equiv 10 \pmod{13}$

Example 73. Let $p, q > 3$ be distinct primes.

   (a) Show that $x^2 \equiv 9 \pmod p$ has exactly two solutions (i.e. ± 3).

   (b) Show that $x^2 \equiv 9 \pmod{pq}$ has exactly four solutions (± 3 and two more solutions $\pm a$).

Solution.

   (a) If $x^2 \equiv 9 \pmod p$, then $0 \equiv x^2 - 9 = (x-3)(x+3) \pmod p$. Since p is a prime it follows that $x - 3 \equiv 0 \pmod p$ or $x + 3 \equiv 0 \pmod p$. That is, $x \equiv \pm 3 \pmod p$.

   (b) By the CRT, we have $x^2 \equiv 9 \pmod{pq}$ if and only if $x^2 \equiv 9 \pmod p$ and $x^2 \equiv 9 \pmod q$. Hence, $x \equiv \pm 3 \pmod p$ and $x \equiv \pm 3 \pmod q$. These combine in four different ways.

      For instance, $x \equiv 3 \pmod p$ and $x \equiv 3 \pmod q$ combine to $x \equiv 3 \pmod{pq}$. However, $x \equiv 3 \pmod p$ and $x \equiv -3 \pmod q$ combine to something modulo pq which is different from 3 or -3.

Why primes >3? Why did we exclude the primes 2 and 3 in this discussion?

Comment. There is nothing special about 9. The same is true for $x^2 \equiv a^2 \pmod{pq}$ for each integer a.

Example 74. Determine all solutions to $x^2 \equiv 9 \pmod{35}$.

Solution. By the CRT:

$$x^2 \equiv 9 \pmod{35}$$
$$\iff x^2 \equiv 9 \pmod 5 \text{ and } x^2 \equiv 9 \pmod 7$$
$$\iff x \equiv \pm 3 \pmod 5 \text{ and } x \equiv \pm 3 \pmod 7$$

The two obvious solutions modulo 35 are ± 3. To get one of the two additional solutions, we solve $x \equiv 3 \pmod 5$, $x \equiv -3 \pmod 7$. [Then the other additional solution is the negative of that.]

$x \equiv 3 \cdot 7 \cdot \underbrace{7^{-1}_{\bmod 5}}_{3} - 3 \cdot 5 \cdot \underbrace{5^{-1}_{\bmod 7}}_{3} \equiv 63 - 45 \equiv 18 \pmod{35}$

Hence, the solutions are $x \equiv \pm 3 \pmod{35}$ and $x \equiv \pm 17 \pmod{35}$. $[\pm 18 \equiv \pm 17 \pmod{35}]$

Silicon slave labor. We can let Sage (more next page!) do the work for us:

```
Sage]  solve_mod(x^2 == 9, 35)
```

   $[(17), (32), (3), (18)]$

Any serious cryptography involves computations that need to be done by a machine. Let us see how to use the open-source computer algebra system **Sage** to do basic computations for us.

Sage is freely available at `sagemath.org`. Instead of installing it locally (it's huge!) we can conveniently use it in the cloud at `cocalc.com` from any browser.

[For basic computations, you can also simply use the textbox on our course website.]

Sage is built as a **Python** library, so any Python code is valid. For starters, we will use it as a fancy calculator.

Example 75. Let's start with some basics.

```
Sage] 17 % 12
```

    5

```
Sage] (1 + 5) % 2   # don't forget the brackets
```

    0

```
Sage] inverse_mod(17, 23)
```

    19

```
Sage] xgcd(17, 23)
```

    $(1, -4, 3)$

```
Sage] -4*17 + 3*23
```

    1

```
Sage] euler_phi(84)
```

    24

Example 76. Why is the following bad?

```
Sage] 3^1003 % 101
```

    27

The reason is that this computes 3^{1003} first, and then reduces that huge number modulo 101:

```
Sage] 3^1003
```

    356959121259817791960422920133078978810663948843080005269528499421243721283610322876 01\
    0144739664176730255639978155597236106757737167167106203642535819647491987457460803546 6\
    1704706398904182050714408540803174892687110481591021823549827662286672460340211243666 8\
    0938796929894977046872005018707156494288273567796241725122202172183616724275431297321 6\
    8010229102922713154530775386398517183447789526555113958789446315044211288493307759874 6\
    041251617347746428658788556867377476037709 0940027

We know how to efficiently avoid computing huge intermediate numbers (binary exponentiation!). Sage does the same if we instead use something like:

```
Sage] power_mod(3, 1003, 101)
```

    27