Exercise 1

a) Find integers u, v such that $36u + 25v = 1$.

b) Find an integer x that satisfies the system of congruences $x \equiv 3 \pmod{36}$ and $x \equiv 12 \pmod{25}$. Then find the set of *all* solutions.

c) Do the same for the system $x \equiv 3 \pmod{36}$ and $x \equiv 12 \pmod{25}$ and $x \equiv 5 \pmod{29}$.

d) Find $m, n \geq 2$ and a, b with $m \neq n$ such that $x \equiv a \pmod{m}$, $x \equiv b \pmod{n}$ has no solutions.

Solution:

a) We run the algorithm from Bézout's lemma:

$$(36, 1, 0),\ (25, 0, 1),\ (11, 1, -1),\ (3, -2, 3),\ (2, 7, -10),\ (1, -9, 13)$$

We can read off that the GCD is indeed 1 (as required) and $u = -9$ and $v = 13$.

b) We have $25 \cdot 36 = 900$. The formula from the lecture gives us, together with the values for u and v from a): $x \equiv 13 \cdot 25 \cdot 3 + (-9) \cdot 36 \cdot 12 \pmod{900}$. Simplifying yields $x \equiv -2913 \pmod{900}$. Thus, -2913 satisfies the two congruences.

If we want, we can reduce -2913 modulo 900 to give us the solution 687. The set of all solutions is precisely the numbers that satisfy $x \equiv 687 \pmod{900}$, i.e. the congruence class $\overline{687}$ mod 900, or more explicitly: $\{687 + 900n \mid n \in \mathbb{Z}\}$

c) In b) we already showed that the two congruences $x \equiv 3 \pmod{36}$ and $x \equiv 12 \pmod{25}$ are equivalent to the single congruence $x \equiv 687 \pmod{900}$. Thus we are now looking for the solutions of the system $x \equiv 687 \pmod{900}$ and $x \equiv 5 \pmod{29}$.

To this end, we run the Euclidean algorithm on 900 and 29:

$$(900, 1, 0),\ (29, 0, 1),\ (1, 1, -31)$$

The GCD is 1 again, so the CRT is applicable. We read off $u = 1$ and $v = -31$ (we can verify this by computing $1 \cdot 900 \cdot -31 \cdot 29 = 1$).

Plugging all the values into the CRT, we find that $x \equiv -31 \cdot 29 \cdot 687 + 1 \cdot 900 \cdot 5 \pmod{900 \cdot 29}$. Simplifying gives us $x \equiv -613113 \pmod{26100}$.

Reducing modulo 26100 gives us $x \equiv 13287 \pmod{26100}$. (Don't worry, the numbers in the exam will *not* be this big.)

d) We can e.g. choose $m = 8$, $n = 4$, $a = 0$, $b = 1$. Then it is easy to see that any number x that satisfies $x \equiv 0 \pmod{8}$ must be a multiple of 8, but no multiple of 8 can possibly satisfy $x \equiv 1 \pmod{4}$ since $8 \cdot k \equiv_4 0 \cdot k \equiv_4 0$.

In fact, any number with $x \equiv a \pmod{8}$ automatically satisfies $x \equiv a \pmod{4}$. And conversely, any number with $x \equiv a \pmod{4}$ automatically satisfies either $x \equiv a \pmod{8}$ or $x \equiv a + 4 \pmod{8}$. This makes it clear that we do not have the same kind of 'independence' as we had for coprime moduli: the CRT essentially says that if we know the value of x modulo m then that tells us nothing about the value of x modulo n (as long as m and n are coprime).

Exercise 2

a) Compute $7^{10002} \bmod 125$.
Hint: This computation is very short if you remember Euler's Theorem and its consequences.

b) Compute $3^{173} \bmod 17$.
Hint: Again use Euler's Theorem, then use fast modular exponentiation.

c) Compute $4320^{12345} \bmod 4321$.
Hint: You do not need to do any arithmetic here if you're clever!

d) Compute $10^{130} \bmod 48$. Note that $\gcd(10, 48) \neq 1$, so you cannot apply Euler's Theorem.
Hint: Determine $10^{130} \bmod 16$ and $10^{130} \bmod 3$ separately and then use the CRT.

Solution:

a) We have a very big exponent, but since $\gcd(7, 125) = 1$ we can save ourselves a lot of work by using Euler's Theorem to reduce the exponent modulo $\varphi(125)$. We have $125 = 5^3$, so $\varphi(125) = 4 \cdot 5^2 = 100$. Thus $7^{10002} \equiv_{125} 7^{10002 \bmod 100} = 7^2 = 49$. In conclusion, $7^{10002} \bmod 125 = 49 \bmod 125$.

b) We have $\gcd(3, 17) = 1$ so Euler's Theorem is applicable and we can reduce the exponent modulo $\varphi(17) = 16$. We have $173 \bmod 16 = 13$, so we must still compute $3^{13} \bmod 17$. We do this as discussed in the lecture, by running the fast exponentiation algorithm in $\mathbb{Z}_{17}$ and reducing all the representatives in our intermediate results modulo 17 after every step.

$$(\overline{3}, 13, \overline{1}) \to (\overline{9}, 6, \overline{3}) \to (\overline{81}, 3, \overline{3}) = (\overline{13}, 3, \overline{3}) \to (\overline{169}, 1, \overline{39}) = (\overline{16}, 3, \overline{5}) \to \overline{16} \cdot \overline{5} = \overline{80} = \overline{12}$$

We thus get the final result that $3^{173} \bmod 17 = 12$.

Alternatively, we could have made our calculation a bit easier by writing e.g. $\overline{-4}$ instead of $\overline{13}$ and $\overline{-1}$ instead of $\overline{16}$. The effect of this is that the numbers we have to multiply are a bit smaller:

$$(\overline{3}, 13, \overline{1}) \to (\overline{9}, 6, \overline{3}) \to (\overline{81}, 3, \overline{3}) = (\overline{-4}, 3, \overline{3}) \to (\overline{16}, 1, \overline{39}) = (\overline{-1}, 3, \overline{5}) \to \overline{-1} \cdot \overline{5} = \overline{-5} = \overline{12}$$

c) $4320^{12345} \equiv_{4321} (-1)^{12345}$. Since $(-1)^n$ is 1 if n is even and -1 otherwise, we get $4320^{12345} \equiv_{4321} -1$, i.e. $4320^{12345} \bmod 4321 = 4320$.

d) We first compute $10^{130} \bmod 3$ by noting that $10^{130} \equiv_3 1^{130} = 1$, i.e. $10^{130} \bmod 3 = 1$.

Next we compute $10^{130} \bmod 16$ by realising that $10^{130} = 2^{130} \cdot 5^{130}$. Clearly, $16 = 2^4$ divides 2^{130} and thereby also 10^{230}, so we have $10^{130} \bmod 16 = 0$.

Let us write $x := 10^{130} \bmod 48$. From what we have computed so far, we know that x satisfies the congruences $x \equiv 0 \pmod{16}$ and $x \equiv 1 \pmod{3}$. The CRT tells us that, since 16 and 3 are coprime, x is uniquely determined by this information (up to a multiple of 48). We can now use the CRT to actually compute x (up to a multiple of 48) from this information.

To this end, we must find u, v such that $16u + 3v = 1$. We could do this using the Extended Euclidean Algorithm, but in this case a keen eye will spot that $u = 1$, $v = -5$ works. Thus via the CRT we have $x \equiv_{48} -5 \cdot 3 \cdot 0 + 1 \cdot 16 \cdot 1 = 16$. In conclusion, $10^{130} \equiv_{48} 16$ and therefore $10^{130} \bmod 48 = 16$.

Exercise 3

a) Find $\varphi(n)$ for $n = 10, 30, 64, 210, 2000$ using the results from the lecture. Also compute $\varphi(n)/n$ to two decimals.

b) For any number n between 2 and 20, compute $\varphi(n)$ and approximate $\varphi(n)/n$ to two decimals. Arrange your results in a table, including those from a).

If you are confident that you can do it by hand but want to save yourself some work, feel free to use your program from d) or some mathematical software of your choice.[1]

c) For which of the n that you considered in a) and b) is $\varphi(n)/n$ particularly big or small? Using your observations, can you find a number n such that $\varphi(n)/n > 0.99$? Can you find one such that $\varphi(n)/n < 0.2$?

d) Write a program that computes $\varphi(n)$ for a given $n \geq 2$. You can compute it directly from its definition (the number of integers from 1 to n coprime to n) or using the prime factorisation.

Solution:

a)
- $\varphi(7) = 6$ (because 7 is prime) and $\varphi(7)/7 \approx 0.86$
- $\varphi(10) = \varphi(2) \cdot \varphi(5) = 1 \cdot 4 = 4$ and $\varphi(10)/10 = 0.40$
- $\varphi(30) = \varphi(2) \cdot \varphi(3) \cdot \varphi(5) = 1 \cdot 2 \cdot 4 = 8$ and $\varphi(30)/30 \approx 0.27$
- $\varphi(64) = \varphi(2^6) = (2-1) \cdot 2^{6-1} = 32$ and $\varphi(64)/64 = 0.50$
- $\varphi(210) = \varphi(2) \cdot \varphi(3) \cdot \varphi(5) \cdot \varphi(7) = 1 \cdot 2 \cdot 4 \cdot 6 = 48$ and $\varphi(210)/210 \approx 0.23$
- $\varphi(2000) = \varphi(2^4) \cdot \varphi(5^3) = 1 \cdot 2^3 \cdot 4 \cdot 5^2 = 800$ and $\varphi(2000)/2000 \approx 0.40$

b)

n	prime decomp.	$\varphi(n)$	$\varphi(n)/n$	n	prime decomp.	$\varphi(n)$	$\varphi(n)/n$
2	2	1	0.50	13	13	12	0.92
3	3	2	0.67	14	$2 \cdot 7$	6	0.43
4	2^2	2	0.50	15	$3 \cdot 5$	8	0.53
5	5	4	0.80	16	2^4	8	0.50
6	$2 \cdot 3$	2	0.33	17	17	16	0.94
7	7	6	0.86	18	$2 \cdot 3^2$	6	0.33
8	2^3	4	0.50	19	19	18	0.95
9	3^2	6	0.67	20	$2^2 \cdot 5$	8	0.40
10	$2 \cdot 5$	4	0.40	64	2^6	32	0.50
11	11	10	0.91	210	$2 \cdot 3 \cdot 5 \cdot 7$	48	0.23
12	$2^2 \cdot 3$	4	0.33	2000	$2^4 \cdot 5^3$	800	0.40

It is apparent that the ratio $\varphi(n)/n$ is big whenever n is a prime, and the bigger n is the bigger the ratio is. Thus, if we want to achieve $\varphi(n)/n > 0.99$ it makes sense to look at the case where n is prime. If n is prime, we have $\varphi(n)/n = (n-1)/n$. Setting $(n-1)/n > 0.99$ and solving for n, we obtain the equivalent condition $n > 100$. Thus any prime above 100 works, e.g. 101 (and indeed, $\varphi(101)/101 \approx 0.9901$).

Conversely, the ratio is small whenever n consists of many different small prime factors, e.g. $6 = 2 \cdot 3$ or $30 = 2 \cdot 3 \cdot 5$ or $210 = 2 \cdot 3 \cdot 5 \cdot 7$. The next number of this pattern is $2310 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$,

[1]All major computer algebra systems support Euler's φ function: Mathematica has `EulerPhi`, SageMath has `euler_phi`, Maxima has `totient`.

but $\varphi(2310)/2310 = 480/2310 \approx 0.208$. The next one is $30030 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 13$ and indeed it has $\varphi(30030)/30030 = 5760/30030 \approx 0.192$.

c) The direct approach is very easy to implement:

```haskell
phi :: Int -> Int
phi n = length [x | x <- [1..n], gcd x n == 1]
```

We can also compute $\varphi(n)$ using the prime factorisation of n. There are Haskell library for factoring integers, but the following naïve trial division approach also works for small integers (for big n it is very hard to compute $\varphi(n)$ anyway):

```haskell
import Data.List (group)

-- Sorted list of the prime factors of n (with repetitions)
primeFactors :: Int -> [Int]
primeFactors n = aux 2 n
  where aux p 1 = []
        aux p n = if n `mod` p == 0 then p : aux p (n `div` p) else aux (p+1) n

-- Prime factorisation of n as a list of (prime, exponent)
primeFactorisation :: Int -> [(Int, Int)]
primeFactorisation = map (\ps -> (head ps, length ps)) . group . primeFactors

phi :: Int -> Int
phi n = product [(p - 1) * p ^ (e - 1) | (p, e) <- primeFactorisation n]
```

Bonus exercise

In the lecture we remarked that RSA decryption works even for messages $\overline{m} \in \mathbb{Z}/n$ that are not coprime to the modulus n. In this exercise, we will prove that this is so.

a) Show that if m is a multiple of p, then $m^a \equiv m \pmod{p}$ for any integer $a > 0$.

b) Show that if $n \geq 1$ and $a \equiv b \pmod{n}$, then also $a \equiv b \pmod{n'}$ for any factor n' of n.

c) Let $n = pq$ where p, q are primes with $p \neq q$. Let $e, d \in \mathbb{Z}$ with $0 < e, d < \varphi(n)$ and $ed \equiv 1 \pmod{\varphi(n)}$. Show that $m^{ed} \equiv m \pmod{p}$ for any $m \in \mathbb{Z}$, even if $\gcd(m, n) \neq 1$.

d) In the same setting as c), conclude that $m^{ed} \bmod n = m$. **Hint:** Use a) to c) and the CRT.

Solution:

a) If m is a multiple of p and $a > 0$, then clearly $p \mid p^a \mid m^a$ and thus $m^a \equiv 0 \pmod{p}$ and of course also $m \equiv 0 \pmod{p}$.

b) $a \equiv b \pmod{n}$ means that $n \mid (a - b)$. Thus for any n' with $n' \mid n$ we have $n' \mid n \mid (a - b)$ by transitivity and thus $a \equiv b \pmod{n'}$.

c) Suppose m and p are coprime. We have $ed \equiv 1 \pmod{\varphi(p)}$ using b) and the fact that $\varphi(p) \mid \varphi(pq)$ since $\varphi(p) = p - 1$ and $\varphi(pq) = (p-1)(q-1)$. Thus by Euler's Theorem, we have $m^{ed} \equiv m \pmod{p}$.

Now suppose m and p are not coprime. Then $p \mid m$ and from a) we have $m^{ed} \equiv m \pmod{p}$ because $ed > 0$.

d) We use c) once to get $m^{ed} \equiv m \pmod{p}$ and again with p and q swapped to obtain $m^{ed} \equiv m \pmod{q}$.

Thus we know that m^{ed} is a solution of the system of congruences $x \equiv m \pmod{p}$ and $x \equiv m \pmod{q}$. Clearly, m is also a solution of this system. Since p and q are two different primes they are coprime, and thus by the CRT the system has a unique solution modulo pq. Therefore, $m^{ed} \equiv m \pmod{pq}$.