MATH 5: HANDOUT 9
POWERS OF 2. BINARY NUMBERS.

Problem: If a certain population of bacteria doubles every day, and right now we have 1 gram of them, how much we will have in 2 days? in a week? in a month?

The answer: after 1 day we would have 2 grams; after 2 days, $2 \times 2 = 4$ grams; ...; after n days, we will have $2 \times 2 \cdots \times 2$ (n times) grams. There is a **special notation** for this:

$$2^n = 2 \times 2 \cdots \times 2 \ (n \text{ times})$$

This grows very fast: for $n = 10$ (in ten days) we will have $2^{10} = 1,024$ grams; in another ten days, the amount will again multiply by 1,024, so we will have $1,024 \times 1,024 \approx 1,000,000$ grams, or one ton of bacteria; in 30 days, we will have about a thousand tons.

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

In some problems, instead of multiplying by 2 every time, we are dividing by 2 every time:

Problem: guess a number between 1-100, asking questions that can only be answered "Yes" or "No".

Solution: the best strategy is doing it so that every question cuts the number of possibilities in half. So the first question should be "Is the number larger than 50?" If the answer is "Yes", we know that the number is between 51–100; if "No", it is between 1–50. Either way, there are now only 50 possible numbers. Next question should again cut the number of possibilities in half.

Hint: what is the maximum number of answers you can get with k questions? Answer 2^k.

In general, if you have numbers $1 - n$ and ask k "Yes" or "No" questions you need $2^k \geq n$, i.e. you need there to be at least one distinct answer for each distinct number.

BINARY NUMBERS

Usual numbers are written in decimal notation, e.g., 351 means $3 \times 100 + 5 \times 10 + 1$

But we can also use powers of 2. For example, we can write a number 26 as $16 + 8 + 2 = 2^4 + 2^3 + 2^1 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1$. Note that the only digits we get are 0 and 1. Thus, we can encode this number by a sequence of digits 1101 (in binary!).

We also discussed that in computers, letters and other symbols are written as sequences of 0 and 1 (bits); since there 2^n such sequences of length n, and there are 26 letters in English alphabet, we need at least 5 bits ($2^5 = 32$) for each English letter. If we want to have lower- and upper-case letters, punctuation, numbers, accented letters such as é, we need more; in real life, people use 8 bits per symbol (called "byte").

The correspondence between actual letters and their codes, i.e. sequences of 0 and 1, is called encoding. In the most common encoding (Latin 1, aka ISO 8859-1) lower case letter "a" has code 01100001.

DIFFERENT BASES

We also touched on other bases. For example, in base 3, we only use digits 0, 1, 2, and they correspond to powers of 3:

$$21021_3 = 2 \times 81 + 1 \times 27 + 0 \times 9 + 2 \times 3 + 1 \times 1 = 250_{10}$$

If the base is larger than 10, then in addition to digits 0 . . . 9 we use letters A, B, etc. For example, in base 16, we use digits 1,...,9 and letters $A = 10, B = 11, C = 12, D = 13, E = 14, F = 15$. The digits correspond to powers of 16:

$$D4B_{16} = D \times 256 + 4 \times 16 + B \times 1 = 13 \times 256 + 4 \times 16 + 11 \times 1 = 3403_{10}.$$