Counting in base 10, 2 and 16

1. Binary Numbers

A super-important fact:

(Nearly all) Computers store all information in the form of binary numbers. Numbers, characters, images, music files --- all of these are stored in a computer using binary numbers.

But ... how? Well, before we get to that, let's study the binary number system.

A binary number is a string of 0's and 1's.

What is the meaning of a binary number?

Let us look at the decimal system first. Consider the number 5281
- There is one "position" or "place" for each power of 10
- The "1" is in the 1's place
- The "8" is in the 10's place
- The "2" is in the 100's place
- The "5" is in the 1000's place

So we have: 5281 = 5 * 1000 + 2 * 100 + 8 * 10 + 1 * 1

Now let's move to the binary number system. Consider this number: 110101. There is a separate "place" for each power of 2. Starting at the right, we have:
- a 1 in the 2^0=1's place
- a 0 in the 2^1=2's place
- a 1 in the 2^2=4's place
- a 0 in the 2^3=8's place
- a 1 in the 2^4=16's place
- a 1 in the 2^5=32's place

So we have: 110101=1*32 + 1*16 + 0*8 + 1*4 + 0*2 + 1 = 53

In other words: 110101 in binary is equivalent to 53 in our usual decimal system of numbers.

A few more examples:

- 1 1 1 1 = 1*8 + 1*4 +1*2 +1*1 = 15
- 0 1 0 1 = 0*8 + 1*4 + 0*2 + 1*1 = 5
- 1 0 1 1 1 = 1*16 + 0*8 + 1*4 + 1*2 + 1*1 = 23
- 1 1 = 1*2 + 1 = 3
- 1 1 1 0 = 1*8+ 1*4 + 1*2 = 14

Some Terminology:
- Each digit of a binary number (each 1 or 0) is called a bit.

- 1 byte = 8 bits.
- 1 KB = 1 kilobyte = 2^10 bytes = 1024 bytes (approx 1 thousand bytes).
- 1 MB = 1 Megabtye = 2^20 bytes = 1,048,580 bytes (approx 1 million bytes).
- 1 GB = 1 Gigabyte = 2^30 bytes = 1,073,741,824 bytes (approx 1 billion bytes)

Converting decimal to binary:

To convert a decimal number to binary, keep dividing the number by 2 until you get down to 0. Keep track of the remainders.

Example: Consider the decimal number 57.

- 57 divided by 2 = 28 Remainder 1
- 28 divided by 2 = 14 Remainder 0
- 14 divided by 2 = 7 Remainder 0
- 7 divided by 2 = 3 Remainder 1
- 3 divided by 2 = 1 Remainder 1
- 1 divided by 2 = 0 Remainder 1

Now list the remainder values from bottom to top: 111001: this is the binary form of 57.

Let us check our answer (111001) by converting it back to decimal:

111001 = 32 + 16 + 8 + 1 = 57, so we are correct.

Maximum Value of a binary number:

Consider a binary number with N bits (where N is a number).
Its maximum possible value is $2^N - 1$ (2 to the power of N, minus 1)
Example:
let N = 3, for a 3-bit binary number, the maximum value is 111, i.e. 2^3-1=7

2. Counting Using Binary Numbers

Consider how counting works in the decimal system. We start with 1 digit. We count using the numerals 0 through 9. After we reach 9, we've run out of numerals. So, we have to add a second digit. We start that digit at 1. Then we cycle the first digit through the numerals 0 through 9 again, to create the numbers 10-19. After we reach 19, we've run out of numerals in the "1's place" again, so we increment the second digit to 2. Eventually, we reach 99. We've run out of numerals in the "1's" place, so we want to increment the second digit again. But, now we've run out of numerals for the second digit as well. So, we have to introduce a 3rd digit, and we start it at 1. And so on.

Counting using binary numbers works the same way, except that we only have 2 numerals (1 and 0) for each digit. So, we start with 1 digit. We count using the numerals 0 through 1:

0
1

We are already out of numerals. So, we have to add a second digit. We start that digit at 1, and then we can cycle the first digit through the numerals 0 through 1 again:

10
11

Next we add a 3rd digit, and start it at 1. Now we can cycle the 1st and 2nd digits as we did before:

100
101
110
111

And so on. Starting from the beginning again, the sequence of binary numbers looks like this:

0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1111
10000
...

Let us rewrite this sequence, with the decimal value of each number listed to the right:

binary	decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

1011	11
1100	12
1101	13
1110	14
1111	15
10000	16
...	

Here is a quick quiz. Consider this large binary number: 101010111. What is the next number?

Answer: 101011000

3. Hexadecimal

We've looked at the decimal number system (base 10) and the binary number system (base 2). The hexadecimal system has a base of 16, but it works the same way.

One tricky point regarding the hexadecimal system is that it uses 16 different numerals: the regular numerals 0 through 9, and then the capital letters A through F.

hexadecimal	decimal equivalent
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Converting Hexadecimal to Decimal

Consider this hexadecimal number: AC7. There is a separate position for each power of 16. Starting at the right, we have:

- a 7 in the 1's place
- a C in the 16's place
- an A in the 16^2=256's place

So we have:

AC7 = A*256 + C*16 + 7*1 = 10*256 + 12*16 + 7 = 2759

In other words: AC7 in hexadecimal is equivalent to 2759 in our usual decimal system of numbers.

Converting Binary to Hexadecimal (Hex)

Consider the following table, which shows the hexadecimal equivalent of the 1st 16 binary numbers.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Given this chart, it is easy to transform binary numbers into decimal form. Let us take an 8-bit example: 10001100

To convert to hex, we divide the number into groups of 4 bits (starting at the right-hand side):

1000 1100

Then, we convert each group of 4 bits to the hexadecimal equivalent, using the chart above:

8 C

And that is all there is to it! Thus, the binary number 10001100 is equal to 8C in hex.

This easy conversion from binary to hexadecimal makes hexadecimal notation very convenient. Instead of writing out a very long binary number, we can represent it compactly using hexadecimal.

Example #2:

Start with this binary number: 100101000110101

Divide the number into groups of 4 bits:

```
100 1010 0011 0101
```

Now add a 0 to the front of the 3-digit group on the left:

```
0100 1010 0011 0101
```

Now convert each 4-digit binary number to a hex number:

```
 4   A    3   5
```

And this is our final answer: 4A35

Converting Hex to Binary

Converting a hexadecimal number to binary is just as easy: just convert each hexadecimal digit to a 4-digit binary number, according to the same table!

Example:

```
D7A =  D     7      A = 1101  0111   1010 = 110101111010
```

Counting in Hexadecimal

Counting in hexadecimal is just like counting in the decimal system, except that we cycle each digit through the numerals 0 to F (instead of 0 to 9).

Thus, counting in hexadecimal goes like this:

hexadecimal	decimal equivalent
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

A	10
B	11
C	12
D	13
E	14
F	15
10	16
11	17
12	18
...etc.	
1F	31
20	32
21	33
...etc.	
2F	47
30	48
31	49
...etc.	
FF	255
100	256

4. The ASCII Code (American Standard Code for Information Interchange)

Computers store all information as numbers (binary numbers). So, how do we represent a keyboard character as a number? The answer is: we use the ASCII Code! This is in fact a very simple idea:
The ASCII Code assigns a number (between 0 and 127 inclusive) to each keyboard character. Each ASCII code number is represented by a 7-bit binary number. However, 8 bits are used for each number, for convenience.

Examples: (The numbers shown are decimal numbers.)

- A to Z: 65 - 90
- 0 to 9: 48 - 57
- $ - 36

Application:

	D	e	a	r
ASCII code	68	101	97	114
Hexadecimal	44	65	61	72
Binary	01000100	01100101	01100001	01110010

5. Unicode

The ASCI code is insufficient: 127 characters is not enough particularly for other languages than English. Unicode was developed to solve this problem. It works under the ame principle as ASCII code, with 1 unique number per character. However, the code supports many more characters (at least 65,536)