WEEK 2

ASCII (American Standard Code for Information Interchange) is a character encoding standard used for representing text in computers. Each character (letter, number, symbol) is assigned a unique 7- or 8-bit binary number, known as its ASCII code.

For example:

The letter a has an ASCII value of 97, which in binary is 01100001.
The letter A (uppercase) has an ASCII value of 65, which in binary is 01000001.

Steps for ASCII ENCODING

1. Convert the letter to Decimal
2. Convert the decimal to binary
3. Join the binary of each letter to encode the ASCII for the word.

Example 1: Encoding a word
Let's go through the encoding for the word "bet" (from the problem):

b (lowercase) has an decimal value of 98, which in binary is 01100010.
e (lowercase) has an decimal value of 101, which in binary is 01100101.
t (lowercase) has an decimal value of 116, which in binary is 01110100.

How to convert a letter to decimal ??

Character	Decimal	Binary	Hexadecimal
A	65	01000001	0x41
B	66	01000010	0x42
a	97	01100001	0x61
b	98	01100010	0x62
1	49	00110001	0x31
@	64	01000000	0x40

Character	Decimal	Hexadecimal	Binary
a	97	0x61	01100001
b	98	0x62	01100010
c	99	0x63	01100011
d	100	0x64	01100100
e	101	0x65	01100101
f	102	0x66	01100110
g	103	0x67	01100111
h	104	0x68	01101000
i	105	0x69	01101001
j	106	0x6A	01101010
k	107	0x6B	01101011
l	108	0x6C	01101100
m	109	0x6D	01101101
n	110	0x6E	01101110
o	111	0x6F	01101111
p	112	0x70	01110000
q	113	0x71	01110001
r	114	0x72	01110010
s	115	0x73	01110011
t	116	0x74	01110100
u	117	0x75	01110101
v	118	0x76	01110110
w	119	0x77	01110111
x	120	0x78	01111000
y	121	0x79	01111001
z	122	7A	01111010

Character	Decimal	Hexadecimal	Binary
A	65	0x41	01000001
B	66	0x42	01000010
C	67	0x43	01000011
D	68	0x44	01000100
E	69	0x45	01000101
F	70	0x46	01000110
G	71	0x47	01000111
H	72	0x48	01001000
I	73	0x49	01001001
J	74	0x4A	01001010
K	75	0x4B	01001011
L	76	0x4C	01001100
M	77	0x4D	01001101
N	78	0x4E	01001110
O	79	0x4F	01001111
P	80	0x50	01010000
Q	81	0x51	01010001
R	82	0x52	01010010
S	83	0x53	01010011
T	84	0x54	01010100
U	85	0x55	01010101
V	86	0x56	01010110
W	87	0x57	01010111
X	88	0x58	01011000
Y	89	0x59	01011001
Z	90	0x5A	01011010

How to convert a decimal to binary ??

1. Use of Power 2

Steps:

Start with the highest power of 2 less than or equal to the number you're converting.
Subtract the value of that power from the number.
Continue with the next highest power, repeating the process until you reach zero.
Write down 1 if you used the power of 2, and 0 if you didn't.

For Example:

Convert decimal **97** to binary:

- The largest power of 2 less than or equal to 97 is $2^6 = 64$. So, subtract 64 from 97: $97 - 64 = 33$.
- The next largest power of 2 less than or equal to 33 is $2^5 = 32$. So, subtract 32 from 33: $33 - 32 = 1$.
- The next largest power of 2 less than or equal to 1 is $2^0 = 1$. So, subtract 1 from 1: $1 - 1 = 0$.
- Now, write `1` for powers of 2 you used, and `0` for powers of 2 you didn't use.

So, 97 in binary is:

- $2^7 = 128 \rightarrow 0$ (since 128 > 97)
- $2^6 = 64 \rightarrow 1$
- $2^5 = 32 \rightarrow 1$
- $2^4 = 16 \rightarrow 0$
- $2^3 = 8 \rightarrow 0$
- $2^2 = 4 \rightarrow 0$
- $2^1 = 2 \rightarrow 0$
- $2^0 = 1 \rightarrow 1$

Thus, $97_{10} = 01100001_2$.

2. Divide by 2 method

Steps:

Divide the decimal number by 2.

Record the remainder (0 or 1).

Divide the quotient by 2 again and repeat until the quotient is 0.

The binary representation is the remainders read in reverse order.

For Example:

Example:

Convert **97** to binary using division by 2:

- $97 \div 2 = 48$ remainder 1
- $48 \div 2 = 24$ remainder 0
- $24 \div 2 = 12$ remainder 0
- $12 \div 2 = 6$ remainder 0
- $6 \div 2 = 3$ remainder 0
- $3 \div 2 = 1$ remainder 1
- $1 \div 2 = 0$ remainder 1

Now, read the remainders from bottom to top:

$97_{10} = 1100001_2$ or, written in 8 bits: 01100001_2.

UNICODE

1. What is Unicode?
Unicode is a standard for encoding characters from almost all writing systems. Each character has a unique code point (a number) like U+543A9, where:

"U+" indicates it's a Unicode code point.
"543A9" is the hexadecimal value representing a specific character.

Steps to convert Unicode into UTF-8 Code:-

- Convert the Unicode code point (in hexadecimal) to binary.
- Determine the number of bytes required based on the length of the binary.
- Use the appropriate format for UTF-8 and fill in the binary bits into the structure.
- Convert each byte into hexadecimal to get the UTF-8 representation.

Step 1: Converting Unicode(in hexadecimal) to binary

Since there are only 16 possible hexadecimal digits (`0` to `F`), you can memorize their binary equivalents in groups. Here's a quick mnemonic grouping for easier recall:

Hex Digit	Binary Equivalent	Memory Tip
0	0000	All zeros
1	0001	One at the end
2	0010	One in the second position
3	0011	Two ones at the end
4	0100	One in the third position
5	0101	4 (0100) + 1
6	0110	4 (0100) + 2
7	0111	4 (0100) + 3 (0011)
8	1000	One in the fourth position
9	1001	8 (1000) + 1
A (10)	1010	8 (1000) + 2
B (11)	1011	8 (1000) + 3
C (12)	1100	8 (1000) + 4 (0100)
D (13)	1101	8 (1000) + 5 (0101)
E (14)	1110	8 (1000) + 6 (0110)
F (15)	1111	All ones

For Example:-

- Hex `B2`:
  - `B` in hex is `1011` in binary.
  - `2` in hex is `0010` in binary.
  - So, `B2` in hex becomes `10110010` in binary.
- Hex `3F`:
  - `3` in hex is `0011` in binary.
  - `F` in hex is `1111` in binary.
  - So, `3F` in hex becomes `00111111` in binary.
- Hex `7A`:
  - `7` in hex is `0111` in binary.
  - `A` in hex is `1010` in binary.
  - So, `7A` in hex becomes `01111010` in binary.

Step 2: Determining number of Bytes required for UTF-8 encoding

Code Point Range	Number of Bytes	Binary Format	Bits Used for Encoding
U+0000 to U+007F	1 byte	`0xxxxxxx`	7 bits
U+0080 to U+07FF	2 bytes	`110xxxxx 10xxxxxx`	11 bits
U+0800 to U+FFFF	3 bytes	`1110xxxx 10xxxxxx 10xxxxxx`	16 bits
U+10000 to U+10FFFF	4 bytes	`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	21 bits

Steps to Determine the Number of Bytes:

Find the length of the binary representation of the Unicode code point (1).

Check which range the binary representation falls into by counting the number of bits.

Determine how many bytes are needed based on the table above.

For Example:-

Example 1: U+543A9

1. **Convert the code point to binary** (from Step 1):
   - `543A9_{16} = 10101000011101010001_2` (21 bits)
2. **Count the number of bits**: There are **21 bits**.
3. **Determine the range**:
   - 21 bits falls into the range `U+10000 to U+10FFFF`, which means we need **4 bytes**.

Example 2: U+0024 (Dollar Sign $)

1. **Convert the code point to binary**:
   - `0024_{16} = 00100100_2` (7 bits)
2. **Count the number of bits**: There are **7 bits**.
3. **Determine the range**:
   - 7 bits falls into the range `U+0000 to U+007F`, which means we need **1 byte**.

Example 3: U+20AC (Euro Sign €)

1. **Convert the code point to binary**:
   - `20AC_{16} = 10000010101100_2` (14 bits)
2. **Count the number of bits**: There are **14 bits**.
3. **Determine the range**:
   - 14 bits falls into the range `U+0800 to U+FFFF`, which means we need **3 bytes**.

Step 3: Construct the UTF-8 Byte sequence

UTF-8 Encoding Patterns:

Number of Bytes	Binary Format	Bits Available for Encoding
1 byte	`0xxxxxxx`	7 bits
2 bytes	`110xxxxx 10xxxxxx`	11 bits
3 bytes	`1110xxxx 10xxxxxx 10xxxxxx`	16 bits
4 bytes	`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	21 bits

Example:-

Example: U+00A9 (Copyright ©)

1. **Binary representation** of U+00A9: `10100011` (9 bits).
2. **UTF-8 format for 2 bytes**: `110xxxxx 10xxxxxx`.
3. **Pad the binary with leading zeros (if needed)** to make 11 bits: `00010100011` (11 bits).
4. **Insert the bits into the UTF-8 format**:
   - First 5 bits go into the first byte: `11000010`
   - Next 6 bits go into the second byte: `10100011`
   - UTF-8: `11000010 10100011`
5. **Final result** (in hexadecimal): `C2 A9`.

So, the UTF-8 encoding for U+00A9 is `C2 A9`.