Representing Letters: ASCII
Representing numbers is great -- but what about words? Can we make sentences with binary data?

- **Key Idea:** Every letter is _____ binary bits.*
  (This means that every letter is _____ hex digits.)

- Global standard called the American Standard Code for Information Interchange (ASCII) is a ___________ ____________ for translating numbers to characters.

ASCII Character Encoding Examples:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Char.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b 0100 0001</td>
<td>0x41</td>
<td>A</td>
</tr>
<tr>
<td>0b 0100 0010</td>
<td>0x42</td>
<td>B</td>
</tr>
<tr>
<td>0b 0100 0011</td>
<td>0x43</td>
<td>C</td>
</tr>
<tr>
<td>0b 0100 0100</td>
<td>0x24</td>
<td>$</td>
</tr>
</tbody>
</table>

Q: Are there going to be any issues with ASCII?

Representing Letters: Other Character Encodings
Since ASCII uses only 8 bits, we are limited to only 256 unique characters. There’s far more than 256 characters -- and what about EMOJIs?? 🎉

- **Many** other character encodings exist other than ASCII.
- The most widely used character encoding is known as Unicode Transformation Format (8-bit) or _______.
- Standard is ISO/IEC 10646 (Latest update is :2002, or v13).

Technical Details of UTF-8 Encoding
UTF-8 uses a ___________ -bit design where each character by be any of the following:

<table>
<thead>
<tr>
<th>Length</th>
<th>Byte #1</th>
<th>Byte #2</th>
<th>Byte #3</th>
<th>Byte #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-byte</td>
<td>0___ ___</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-bytes:</td>
<td>110_ __</td>
<td>10__ ___</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-bytes:</td>
<td>1110 ___</td>
<td>10__ ___</td>
<td>10__ ___</td>
<td></td>
</tr>
<tr>
<td>4-bytes:</td>
<td>1111 0___</td>
<td>10__ ___</td>
<td>10__ ___</td>
<td>10__ ___</td>
</tr>
</tbody>
</table>

Unicode characters are represented by U+## (where ## is the hex value of the character encoding data) and all 1-byte characters match the ASCII character encoding:

- ‘a’ is ASCII _______, or ________.

Example: ε (epsilon) is defined as U+03b5. How do we encode this?

Example: I received the following binary message encoded in UTF-8: 0100 1000 0110 1001 1111 0000 1001 1111 1000 1110 1000 1001

1. What is the hexadecimal representation of this message?
2. What is the **byte length** of this message? _______
3. What is the **character length** of this message? _______
4. What does the message say?
Programming in C
Today, you’ll begin your very first program in C!
- You already know how to program in C++! 🎉
- Programming in C is a simplification of the C++ programming.

1. Program Starting Point of **ALL** C PROGRAMS:

2. Printing Using `printf()` (and include `<stdio.h>`):

```c
#include <stdio.h>

int main() {
    int i = 42;
    char *s = "Hello, world!";
    float f = 3.14;

    printf("%d  %s  %f\n", i, s, f);
    printf("%d\n", s[0]);
    printf("%d\n", s);
    printf("%d\n", f);
    return 0;
}
```

3. Pointers:

4. Heap Memory Allocation:

5. **Strings** – There is no “data type” in C known as a string. Instead, we refer to “C Strings” as a sequence of characters:
- A ‘C string’ is just a character pointer: ________.
- The string continues until it reaches a ________ byte.
- C will automatically include the NULL byte ONLY when using double quotes in your code (not counted as part of the length, but does require memory – extremely tricky!)

---

**02/string.c**

```c
char *s = malloc(6);
strcpy(s, "cs240");
printf("\s[0]\: 0x%x == %d == %c\n", s[0], s[0], s[0]);
printf("\s[4]\: 0x%x == %d == %c\n", s[4], s[4], s[4]);
printf("\s[5]\: 0x%x == %d == %c\n", s[5], s[5], s[5]);
printf("s == "%s", strlen(s): %ld\n\n", s, strlen(s));
char *s2 = s + 2;
printf("\s2[0]\: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == "%s", strlen(s2): %ld\n\n", s2, strlen(s2));
*s2 = 0;
printf("\s2[0]\: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == "%s", strlen(s2): %ld\n\n", s2, strlen(s2));
printf("s == "%s", strlen(s): %ld\n", s, strlen(s));
```

---

**02/utf8.c**

```c
char *s = malloc(5);
s[0]=0xF0; s[1]=0x9F; s[2]=0x8E; s[3]=0x89; s[4]=0x00;
s1 = \xF0\x9F\x8E\x89;
s2 = "🎉";
s3 = \U0001f389"; // \U - must be 8 bytes

printf("s s s %s", s, s1, s2, s3);
printf("strlen(): %ld %ld %ld %ld\n", strlen(s), strlen(s1), strlen(s2), strlen(s3));
```