19.- ft_memcmp.-

Function based on the definition given in the BSD man pages for “memcmp(3)”.
The library associated is <string.h> (standard C library).

Synopsis: `int memcmp(const void *s1, const void *s2, size_t n);`

Purpose:

Compares two memory blocks (`s1` and `s2`) byte by byte up to a specified number of bytes (`n`).

Parameters:

- `s1`: The first memory block to compare.
- `s2`: The second memory block to compare.
- `n`: The maximum number of bytes to compare.

Return Value:

Returns an integer representing the difference between the first non-matching bytes:

- 0 if the blocks are equal up to `n` bytes.
- A negative value if `s1` is lexicographically less than `s2`.
- A positive value if `s1` is lexicographically greater than `s2`.

Description:

- Iterates through the bytes of `s1` and `s2` simultaneously, comparing them until a difference is found or `n` bytes have been compared.
- Returns the difference between the first non-matching bytes.

Code:

```
#include "libft.h"

int ft_memcmp(const void *s1, const void *s2, size_t n)
{
    unsigned const char *str1 = s1;
    unsigned const char *str2 = s2;
    size_t i = 0;

    while (i < n)
    {
        if (str1[i] != str2[i])
            return (str1[i] - str2[i]);
        i++;
    }
    return (0);
}
```

Code Explanation:

1. **Cast to unsigned char:** Converts `s1` and `s2` to `unsigned char` pointers for byte-level comparison.
2. **Iterates through bytes:**
    - Continues as long as `i` is less than `n` (the specified comparison limit).

    - Compares the current bytes at `str1[i]` and `str2[i]`.
3. **Returns difference:**
    - If a difference is found, returns the difference between the byte values (cast to `int`).
4. **Returns 0:** If no difference is found within `n` bytes, returns 0 (blocks are considered equal).

Key Points:

- **Memory Block Comparison:** Works with any memory blocks, not just strings.
- **Byte-Level Comparison:** Compares bytes directly for efficient comparison.
- **Maximum Comparison Limit:** Compares up to `n` bytes, even if the memory blocks are larger.
- **Character Casting:** Uses `(unsigned char)` for proper comparison of characters, including extended ASCII values.
- **Lexicographical Order:** Compares bytes based on their ASCII values, determining which block is "less than" or "greater than" the other.