20.- ft_strnstr.-

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

Synopsis:

```
char *strnstr(const char *haystack, const char *needle, size_t len);
```

Purpose:

Locates the first occurrence of a substring (`needle`) within a larger string (`haystack`), searching up to a specified number of characters (`len`).

Parameters:

- `haystack`: The string to search within.
- `needle`: The substring to search for.
- `len`: The maximum number of characters to search within `haystack`.

Return Value:

Returns a pointer to the first occurrence of `needle` within `haystack`, or `NULL` if `needle` is not found within the first `len` characters of `haystack`.

Description:

- Iterates through `haystack` up to `len` characters, comparing substrings with `needle` until a match is found or the limit is reached.
- Returns a pointer to the beginning of the matching substring if found.

Code:

```
#include "libft.h"

char *ft_strnstr(const char *haystack, const char *needle, size_t len)
{
    size_t i = 0;
    size_t j;

    if (needle[0] == '\0')  // Handle empty needle case
        return ((char *)haystack);
    while (i < len && haystack[i])
    {
        j = 0;
        while (needle[j] && haystack[i + j] == needle[j] && i + j < len)
            j++;
        if (needle[j] == '\0')  // Match found
            return ((char *)&haystack[i]);
        i++;
    }
    return (NULL);  // Not found
}
```

Code Explanation:

1. **Handles empty needle:** If `needle` is empty, returns `haystack` immediately.
2. **Iterates through haystack:**

    - Continues as long as `i` is less than `len` and `haystack[i]` is not the null terminator.
3. **Compares substrings:**
    - Iterates through `needle` (`j` loop) to compare characters with `haystack` at `i + j`.
    - Ensures comparison stays within `len` limit (`i + j < len`).
4. **Returns match:**
    - If the end of `needle` is reached (`needle[j] == '\0'`), a match is found, and the function returns a pointer to the beginning of the match in `haystack`.
5. **Returns NULL:** If no match is found within `len` characters, returns `NULL`.

Key Points:

- **Limited Search:** Searches up to `len` characters, even if `haystack` is longer.
- **Substring Search:** Specifically designed for finding substrings within strings.
- **Empty Needle Handling:** Returns the entire `haystack` if `needle` is empty.

Comments for the `main` function:

```
int main(void)
{
    // Define strings to search
    char haystack[] = "Nora zoaz zu";
    char needle[] = "oaz";

    // Call ft_strnstr to find the substring
    char *result = ft_strnstr(haystack, needle, strlen(haystack));

    // Check if the substring was found and print messages
    if (result) {
        printf("ft_strnstr() has found '%s' in string '%s'.\n", needle,
haystack);
        printf("Substring position is %lu in the string.\n", (result - haystack)
+ 1);
        printf("Result is %s", result);
    } else {
        printf("ft_strnstr() unable to find '%s' in string '%s'.\n", needle,
haystack);
    }

    return (0);

}
```