23.- ft_strdup.-

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

Synopsis: `char strdup(const char *s1);`

Purpose:

> Allocates memory for a new string and duplicates the contents of a given string (`src`) into it.

Parameters:

- `s1`: The string to be duplicated.

Return Value:

> Returns a pointer to the newly allocated and duplicated string, or `NULL` if memory allocation fails.

Description:

- Allocates enough memory to hold the string `src` plus the null terminator.
- Copies the contents of `src` to the newly allocated memory using `ft_memcpy`.
- Returns a pointer to the duplicated string.

Code:

```
#include "libft.h"

char *ft_strdup(const char *src)
{
    char *dest;
    size_t len;

    // Calculate length of source string
    len = ft_strlen(src);

    // Allocate memory for the duplicate string
    dest = (char *)malloc(len + 1);  // Allocate space for null terminator

    // Check for allocation failure
    if (dest == NULL)
        return (NULL);

    // Copy the source string to the destination
    ft_memcpy(dest, src, len + 1);  // Copy including null terminator

    return (dest);  // Return pointer to the duplicated string
}
```

Code Explanation:

1. **Calculate Length:** Calls `ft_strlen` to determine the length of `src`.
2. **Allocate Memory:** Allocates `len + 1` bytes (to accommodate the null terminator) using `malloc`.

3. **Check for Allocation Failure:** If `malloc` returns `NULL`, indicates allocation failure and returns `NULL`.
4. **Copy String:** Uses `ft_memcpy` to copy the contents of `src` (including the null terminator) to `dest`.
5. **Return Duplicate:** Returns the pointer `dest` to the newly allocated and duplicated string.

Comments for the `main` Function:

```
int main(void)
{
    // Define an original string
    char *str1 = "Hello my friend";

    // Duplicate the string using ft_strdup
    char *str2 = ft_strdup(str1);

    // Check if duplication succeeded
    if (!str2) {
        printf("Error: Memory allocation failed.\n");
        return (1);
    }

    // Print the duplicated string
    printf("%s\n", str2);

    // Free the allocated memory
    free(str2);  // Important to release memory

    return (0);  // Indicate successful program termination
}
```

Key Points:

- **Dynamic Memory Allocation:** Uses `malloc` to allocate memory for the duplicate string.
- **String Copying:** Uses `ft_memcpy` to efficiently copy the string contents.
- **Null Terminator Handling:** Ensures the null terminator is copied to the duplicate string.
- **Memory Management:** Requires manual memory release using `free` after use.