NAME

mystrcmp - compare two strings
mystrdup - make a duplicate of a string

SYNOPSIS

```
#include "mystring.h"
#include <stdlib.h>
#include <errno.h>

int mystrcmp(const char *s1, const char *s2);

char *mystrdup(const char *s);
```

DESCRIPTION

The mystrcmp() function returns a negative integer, a zero, or a positive integer for situations that ***s1*** is less than, equal to, or greater than ***s2***.

The mystrdup() function returns a pointer pointing to a new string duplicated from original string ***s***. The memory is allocated using malloc(size + 1), and can be freed using free() function, visit http://www.kernel.org/doc/ for more details.

RETURN VALUE

On success, mystrcmp() function returns an integer to indicate whether ***s1*** is greater than ***s2*** or not. Return value will be less than 0 when ***s1*** is less than ***s2***, equal to 0 when ***s1*** is the same as ***s2***, or greater than 0 when ***s1*** is greater than ***s2***.

On success, mystrdup() function returns a pointer pointing to the new duplicated string of original string ***s***. The function will return **NULL** when no sufficient memory is available, with ***errno*** set.

ERRORS

mystrdup() function will set ***errno*** to **ENOMEM** if insufficient memory available when allocating new string.

SOURCE CODE

```
/*
 * to implement strcmp, tried to get all possible return values
 * but compiler would have optimization when inputs are two
constant strings, both in gcc and clang
 * get the index of first different char, using while loop,
break when string get to the end
 * simply return the difference of two char ascii codes
 */
int mystrcmp(const char *s1, const char *s2) {
    int i = 0;
    while (s1[i] == s2[i]) {
        i++;
        if (s1[i] == '\0') {
            break;
        }
    }
    return s1[i]-s2[i];
}
/*
 * first get the size or length of the string, then malloc
 * according manual, malloc(size+1)
 * if malloc failed, return null, may need casting to (char *)
 * if not, define another ptr pointing to the memory
 * set the values: char, add \0 to the end
 */
char *mystrdup(const char *s) {
    int size = 0;
    char *dup;
    while (s[size]) {
        size++;
    }
    dup = (char *)malloc(size * sizeof(char) + 1);
    if (!dup) {
        errno = ENOMEM;
        return (char *)NULL;
    }
    char *set = dup;
    while(*s != '\0') {
        *set = *s;
        set++;
        s++;
    }
    *set = '\0';
    return dup;
}
```