## NAME

mystrcmp - compare two strings
mystrdup – duplicate a specific number of bytes from a string

## SYNOPSIS

```
#include <string.h>
#include “mystring.h”
#include <stdlib.h>
int mystrcmp(const char *s1, const char *s2);
char *mystrdup(const char *s);
```

## DESCRIPTION

The ***mystrcmp()*** function compares the two strings *s1* and *s2*. The locale is not taken into account. It returns an integer less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, to match, or be greater than *s2*. If the s1 and s2 are not equal, then return unsigned integer of difference of ASCII Codes of s1 and s2; otherwise, return 0 in order to show that s1 and s2 are equal.

The **my*strdup*()** function shall return a pointer to a duplicate of the string pointed to by *s*. The returned pointer can be passed to *free*(). A null pointer is returned if the new string cannot be created.

## RETURN VALUE

The ***mystrcmp()***functions return an integer less than, equal to, or greater than zero if *s1* (or the first *n* bytes thereof)is found, respectively, to be less than, to match, or be greater than *s2*.

The ***mystrdup()*** function shall return a pointer to a new string on success. Otherwise, it shall return a null pointer.

**ERRORS**

***mystrcmp()* -** No errors are defined.

***mystrdup()* -** Storage space available is insufficient in order to allocate the memory.

**SOURCE CODE**
```
#include <string.h>
#include "mystring.h"
#include "stdlib.h"
int mystrcmp(const char *s1, const char *s2) {

// Since we are comparing two strings s1 and s2 character by
character
// using a while loop to go through each string
// stop until one of them run to the end
// That means we counter a null character '\0'
  while ((*s1 != '\0') && (*s2 != '\0')) {

// check each charater of two strings one by one
// if the ascii code of current character in first string
// is greater than the code of current character in the second
string
// return f1 - f2 > 0
// if they first one smaller than second one
// return f1 - f2 < 0
// else continue iteration
// By the way casting (const unsigned char*) make sure that
// we will deal with the positive ascii code of characters.

    if (*s1 != *s2) {
      return *(const unsigned char*)s1 - *(const unsigned
char*)s2;
    }

// update the current s1 and s2 pointer
// In other word, let the s1 and s2 pointer point to
// the next character in their strings
    s1++;
    s2++;
  }

// if one of string run to the end then do the last difference
// in order to get the result
  if (*s1 != *s2) {
    return *(const unsigned char*)s1 - *(const unsigned
char*)s2;
  }
// after checking all the character in those two strings
// we can return 0 to show their are equal;
  return 0;
}

char *mystrdup(const char *s) {

// variable i is the index of the newString
// *newString is the new string's pointer which
// point the head of string
// *t is the temporary const char
// which is used for counting the length of input string
// length is the length of the input string
//
  int i = 0;
  char *newString = NULL;
  int length = 1;
  const char *t = s;

// using while loop to count the number of characters in the
// string and stored it in the length variable
  while (*t != '\0') {
    t++;
    length++;
  }

// allocate dynamic memory for the duplicated string (newString)
  newString = (char*)malloc(length);

// go through string character by character and copy each
character
// into newString
  while (*s != '\0') {
    newString[i] = *s;
    s++;
    i++;
  }

// put null terminator at the end of new String then return
// the duplicated string
  newString[i] = '\0';
  return newString;
}
```