C291 C/Unix

Quiz 1 (100 pts total)

In class, we have been learning about how to work with C pointers through looking at string manipulation examples. In class today, you will be writing a program to compare two strings. You function, mystrcnmp, will mirror the functionality of the libc string function strncmp.

The prototype for your function should look like the following:

int strncmp(char *s1, char *s2, int n);

s1 and s2 are pointers to strings and n is the maximum number of characters that should be considered. Your function should compare the two strings character by character and adhere to the following rules:

1.) Compare the first character in s1 and s2. If they are equal, move to the next character. Continue to compare characters until the difference is not zero, you reach the end of one of the strings, or you have compared n characters.

2.) If the characters are not equal, return the difference. The characters in a string are typically represented by the ASCII character set (see attached ASCII chart). Each printable character is assigned a number. Your function should return the difference between the specific character in each string which is different.

For example, if s1 is pointing to an 'A' and s2 is pointing to a ' ' (SP or space character), the value that is returned would be 'A' - ' ' or 65-32=33.

3.) If you reach the end of one of the strings or have compared n characters, the function should return the difference between the last two characters compared.

Examples:

Example 1

```
char s1[] = "hello world";
char s2[] = "hello !";
printf("%d\n",mystrncmp(s1,s2,30));
```

Output

```
86
```

Example 2

```
char s1[] = "hello world";
char s2[] = "hello !";
printf("%d\n",mystrncmp(s1,s2,5));
```

Output

```
0
```

Example 3

```
char s1[] = "hello world";
char s2[] = "hello world ";
printf("%d\n",mystrncmp(s1,s2,30));
```

Output

```
0
```

Example 4

```
char s1[] = "hello world";
char s2[] = "hello world!";
printf("%d\n",mystrncmp(s1,s2,30));
```

Output

```
-33
```

Linux Man Page for strncmp
(only focus on strncmp for this quiz, you can also ignore the "const modifier on s1 and s2. The type size_t on the variable n resolves to the type int)

```
STRCMP(3)              Linux Programmer's Manual              STRCMP(3)

NAME
      strcmp, strncmp - compare two strings

SYNOPSIS
      #include <string.h>

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

      int strncmp(const char *s1, const char *s2, size_t n);

DESCRIPTION
      The  strcmp()  function compares the two strings s1 and s2.  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.

      The  strncmp()  function  is similar, except it compares the only first
      (at most) n bytes of s1 and s2.

RETURN VALUE
      The strcmp() and strncmp() 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.
```

C291 C/Unix

Extra Credit (10 pts)

Write a function called **substring** using your mystrncmp and strlen. This function returns a pointer to the first occurrence of the character substring in the larger string. If the substring is not part of the larger string, return 0;

```
char * substring(char * string, char *substring); // prototype for substring
```

Examples

Example 1

```
char s1[] = "hello world";
char s2[] = "rl";
char * p;
p = substring(s1,s2)
if (p) {
   printf("%c\n",*p);
}
else {
   printf(“Not Found\n”);
}
```

Output

```
r
```

Example 2

```
char s1[] = "hello world";
char s2[] = "foo";
char * p;
p = substring(s1,s2)
if (p) {
   printf("%c\n",*p);
}
else {
   printf(“Not Found\n”);
}
```

Output

```
Not Found
```