Programming Assignment – String Manipulation

In this assignment you will need to create a file named strings.asm with the subroutines SWITCHCASE and COUNTCHAR as summarized below.

SWITCHCASE – Switch the case of all alphabetic characters in a string. Make all uppercase to lowercase and all lowercase to uppercase.
COUNTCHAR – Search a given string for a specific character and return the number of times it occurs in R0.

Processing strings in the LC3.

Just like in C, we have pointers to strings.

LEA loads an address into a register making the register a pointer to the string.

So, something like LEA R1, STR1 will load the address of the first character of STR1 into R1. In other words, R1 is now a pointer to STR1.

Assuming R0 has the address, and RX is any register you want to use.

Note that the code below acts EXACTLY like a while loop with the top two instructions determining if R0 is pointing at a zero and branching out of the loop is if is a zero.

```
SWITCHCASE
      ;Save registers of course
TOP
      LDR RX, R0, #0  ;  RX <- *R0 , RX gets the character pointed at by R0.
      BRZ DONE ; If RX is loaded with a zero, then we are done processing.

      ;Do your work on the character.

      ADD R0, R0, #1 ; Point R0 at the next character.
      BRNZP TOP
DONE
      ;Restore registers of course
      RET
```

ALL OF YOUR FUNCTIONS IN THIS ASSIGNMENT SHOULD DO SOMETHING SIMILAR TO THE ABOVE.

DO NOT FORGET ABOUT SAVING AND RESTORING REGISTERS.

- **EACH SUBROUTINE SHOULD HAVE ITS OWN SEPARATE STORAGE VARIABLES.**
- **EACH SUBROUTINE MUST SAVE ALL REGISTERS THAT IT MODIFIES.**
- **EACH SUBROUTINE MUST RESTORE ALL MODIFIED REGISTERS BEFORE RETURNING.**
- **IF YOU USE A TRAP OR CALL A SUBROUTINE YOU MUST ALSO SAVE REGISTER AND RESTORE R7.**
- **DO NOT SAVE AND RESTORE A DESIGNATED RETURN REGISTER.**

SWITCHCASE – Convert a string's uppercase characters to lowercase and lowercase characters to uppercase.

1. The address of the string should be passed in R0.
2. Convert all the lowercase characters a-z to upper case and all uppercase characters A-Z to lowercase.
    a. Look at each character
    b. If that character is in the range of 'a' to 'z' subtract 32 to make it uppercase.
    c. Else if that character is in the range of 'A' to 'Z' add 32 to make it uppercase.
    d. Continue until you find the end of the string. Remember it will be a zero, not ASCII zero but a real zero.
3. Do not modify any other characters (punctuation, numbers, etc…).
4. Use the following as a "main" for testing. I am using the word "main" to refer to the code at the top of your program that is outside any subroutines. It is just below .orig and just above HALT.

YOU MUST REPLACE THIS MAIN WITH THE ONE LISTED IN THE SUBMITTING SECTION BEFORE SUBMITTING TO WEB_CAT!

```
        ;The following should print:  ABcdEFg5
        .orig x3000
        LEA R0, STR1   ;R0 points to STR1. DELETE THIS LINE BEFORE UPLOADING.
        JSR SWITCHCASE ;Swap the case of all letters in STR1
        PUTS           ;Should print ABcdEFg5
        HALT

        ;DATA SECTION DELETE THE NEXT TWO LINES BEFORE UPLOADING
        .blkw 100 ;Change this value when testing between 50 and 150.
STR1    .stringz "abCDefG5" ;Use numbers, punctuation, spaces, etc... while testing.

    ;************************SUBROUTINES BELOW THIS LINE*******************************
```

COUNTCHAR – Search a string for a specified character. Count and return the number of times that character occurs in the string. You should count uppercase and lowercase versions of that character.

1. R0 should contain the ASCII code of a character to find. R1 should contain a pointer to the string to search for the character.
2. Look at each character in turn starting with the first.
3. If the character is zero branch to the end.
4. If the character is the lowercase version of the character in R0, count the character.
5. If the character is the uppercase version of the character in R0, count the character.
6. Repeat the above steps starting at 3.

7. Use the following as a "main" for testing. I am using the word "main" to refer to the code at the top of your program that is outside any subroutines. It is just below .orig and just above HALT.

   YOU MUST REPLACE THIS MAIN WITH THE ONE LISTED IN THE SUBMITTING SECTION BEFORE SUBMITTING TO WEB_CAT!

```
    ;This example should result in 4 stored in R0.
    .orig x3000
    LD R0, CHR     ;Load character into R0.
    LEA R1, STR1   ;R1 points to STR1
    JSR COUNTCHAR  ;Count the char given in R0 in STR1. R0 should contain the number.
                   ;DO NOT CALL COUNTCHAR TWICE!!!
    HALT           ;COUNTCHAR should return with 4 in R0.

    ;DATA SECTION DELETE THE NEXT FOUR LINES BEFORE UPLOADING
    ;Make sure to test for a character not in STR1.
    .blkw 100  ;Change this value when testing. Vary from 50 to 150.
STR1 .stringz "zkfazj;iozJkfJJz"  ;And, of course, change the string.
CHR .fill 'z' ; And, also of course, change the search char.

; ************************SUBROUTINES BELOW THIS LINE*******************************
```

Submitting

You will be submitting each subroutine above separately on Web-CAT. You can have all subroutines in one file, but you will have to change the "main" section for each submission.

Submitting SWITCHCASE

1. Use the following code as main.

```
.orig x3000
JSR SWITCHCASE ; NOTE ONLY CALL SWITCHCASE ONCE
PUTS
HALT
; ************************SUBROUTINES BELOW THIS LINE*******************************
```

2. This main and your subroutine is ALL that should be included.

   MAKE SURE YOU DELETE THE DATA SECTION.

3. Save the file as strings.asm. Do not load R0. My tests will insert proper values.
4. Upload strings.asm file to Web-CAT .

Submitting COUNTCHAR

1. Use the following code as main. Do not modify or load R0 or R1. My test will insert proper values.

```
.orig x3000
JSR COUNTCHAR      ; ONLY CALL COUNTCHAR ONCE WHEN SUBMITTING!!
HALT
; ************************SUBROUTINES BELOW THIS LINE*******************************
```

2. This main and your subroutine is ALL that should be included.

   MAKE SURE YOU DELETE THE DATA SECTION.

3. Save the file as strings.asm. Do not modify or load R0 or R1. My test will insert the proper values.
4. Upload strings.asm file to Web-CAT.

Here is an ASCII chart for reference.

Dec	Hex	Char
0	00	Null
1	01	Start of heading
2	02	Start of text
3	03	End of text
4	04	End of transmission
5	05	Enquiry
6	06	Acknowledge
7	07	Audible bell
8	08	Backspace
9	09	Horizontal tab
10	0A	Line feed
11	0B	Vertical tab
12	0C	Form feed
13	0D	Carriage return
14	0E	Shift out
15	0F	Shift in
16	10	Data link escape
17	11	Device control 1
18	12	Device control 2
19	13	Device control 3
20	14	Device control 4
21	15	Neg. acknowledge
22	16	Synchronous idle
23	17	End tran. Block
24	18	Cancel
25	19	End of medium
26	1A	Substitution
27	1B	Escape
28	1C	File separator
29	1D	Group separator
30	1E	Record separator
31	1F	Unit separator

Dec	Hex	Char
32	20	Space
33	21	!
34	22	“
35	23	#
36	24	$
37	25	%
38	26	&
39	27	‘
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2c	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	K
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	Delete