**COMP.2030 HW 2: CharType Due Date: 9/27 (Fri) 11:59 PM**

Write a MIPS program which repeatedly read one line at a time from the keyboard. And print types of individual characters of an input line. Character types are given by

Characters Type Comment

0 1 .. 9 1 Digits

A B .. Z a b .. z 2 Letters

\* + - / 3 Operators

. ( ) , : 4 Delimiters

b 5 blank

# 6 End of the Line

Your program should

(1) reads a line from the keyboard,

(2) for each character in the input line, search Tabchar for its type and store types of individual characters on the line,

and (3) print char types of the input line.

This process is repeated for each input line. Namely, your program should be able to repeatedly read an input line and print char types.

As an example, when the following is entered in the keyboard

THISLOOP: LWU R2, 3 #

the output from the program shall be

222222224 222 2141 6

Note that each input line includes ‘#’ as the end-of-the-line symbol, although it may be missing. The blank type of 5 can be left out when printing character types.

**Approaches:**

A rough structure of the program is as follows:

while (1){

getline();

if (inBuf[0] == ‘#’) break;

for (i=0; i<80; i++){

key = inBuf[i];

chType = lin\_search(key);

outBuf[i] = char(chType)

if (key == ‘#’) break;

}

print inBuf

print outBuf

clear inBuf

clear outBuf

}

1. Reading input lines

Besides Tabchar table, buffers to store an input line and outputcharacter types are to be declared.

.data

inBuf: .space 80 # input line

outBuf: .space 80 # char types for the input line

prompt: .asciiz "Enter a new input line. \n”

A procedure call to getline is made by the following MIPS statement:

jal getline

An example of getline procedure to read an input string is as follows.

.text

getline:

la $a0, prompt # Prompt to enter a new line

li $v0, 4

syscall

la $a0, inBuf # read a new line

li $a1, 80

li $v0, 8

syscall

jr $ra

2. Linear search of a character

A simple approach to finding a character type is to arrange all characters and their types into an array. This makes future changes and updates to the character set a quick process. Use the MIPS table Tabchar below and write a linear search program to perform a search for an input character from the input string. Be careful that a character from the input string is stored in a byte where as the search table below is organized in units of words (4-byte each). The linear search part in your program HAS TO BE coded as a function. When you compare a letter from the input string to characters in Tabchar, make it sure that you use ‘**lb**’ instruction to move only a byte out of the input string to one of registers.

**What to submit:**

* Name your MIPS source code starting with your last name and submit the asm file.
* Include sufficient comments in the source code (check HW grading policy below).
* You may alternatively implement the binary search algorithm instead of the linear search.

Notes:

1. From the input keyboard, you can enter the maximum of 80 characters. You can reserve a space of 80 characters by

inBuf: .space 80

1. The end of the input string can be detected by ‘#,’ a Line-Feed, or the first NULL character (0x00).

.data

Tabchar: .word 0x0a, 6 # LF

.word ' ', 5

.word ‘#’, 6

.word ‘$’,4

.word '(', 4

.word ')', 4

.word '\*', 3

.word '+', 3

.word ',', 4

.word '-', 3

.word '.', 4

.word '/', 3

.word '0', 1

.word '1', 1

.word '2', 1

.word '3', 1

.word '4', 1

.word '5', 1

.word '6', 1

.word '7', 1

.word '8', 1

.word '9', 1

.word ':', 4

.word 'A', 2

.word 'B', 2

.word 'C', 2

.word 'D', 2

.word 'E', 2

.word 'F', 2

.word 'G', 2

.word 'H', 2

.word 'I', 2

.word 'J', 2

.word 'K', 2

.word 'L', 2

.word 'M', 2

.word 'N', 2

.word 'O', 2

.word 'P', 2

.word 'Q', 2

.word 'R', 2

.word 'S', 2

.word 'T', 2

.word 'U', 2

.word 'V', 2

.word 'W', 2

.word 'X', 2

.word 'Y', 2

.word 'Z', 2

.word 'a', 2

.word 'b', 2

.word 'c', 2

.word 'd', 2

.word 'e', 2

.word 'f', 2

.word 'g', 2

.word 'h', 2

.word 'i', 2

.word 'j', 2

.word 'k', 2

.word 'l', 2

.word 'm', 2

.word 'n', 2

.word 'o', 2

.word 'p', 2

.word 'q', 2

.word 'r', 2

.word 's', 2

.word 't', 2

.word 'u', 2

.word 'v', 2

.word 'w', 2

.word 'x', 2

.word 'y', 2

.word 'z', 2

.word 0x5c, -1 # if you ‘\’ as the end-of-table symbol

**HW Grading Policy**

HWs will be graded according to the following distribution:

Correctness of execution (60%), Comments (40%).

* Comments are critically important, and comments will be graded.
* Each function has to be preceded by comments including the name of the function, arguments and their registers, and how the return value is passed.
* Each line of MIPS instruction is typically commented on the right in a short C-like statement, so that comments on the right provide enough info on the flow of the function
* Don’t just copy/paste and submit the incomplete homework code from class. That’s essentially not passing in anything, so you will be getting a 0.
* Read the homework assignments carefully. If you miss any of the requirements I will take points off (yes, even if it compiles fine).
* No plagiarizing. I will give you a 0 and report you if I catch you copying from one another/cheating.
* Working with your classmates on homework is fine, but you have to write your code and are not allowed to copy off one another (plagiarizing). As such, please make sure all your work that you have completed is written by you and you alone. The logic and reasoning can be the same, but not the code. I understand if certain sections can only be written in one way, but it does look very fishy if you and another student’s HW look exactly the same. Please write the names of the people you worked with on the top comment of the code to clear up any confusion.