**COMP.2030 HW 4: CharType**

Write a MIPS program which repeatedly reads a line of characters from the keyboard, determines the “type” of each character in that line, and then prints the sequence of character types matching the input string. The different character types are:

Characters Type Comment

0 1 .. 9 1 Digits

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

\* + - / 3 Operators

. ( ) , : $ 4 Delimiters

# 5 End of the Line

6 blank (0x20)

Anything else 0

Your program should repeat the following operations in a loop:

1. read a line from the keyboard and save it in a buffer inBuf of 80 characters,
2. for each character in the input line, search the table tabChar for its type and save the corresponding numeric type value in another buffer outBuf,
3. print the outBuf that was filled for this input line, and
4. repeat back at step 1

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

THISLOOP: LWU R2, 3 #

the output printed by the program should be

222222224 222 2141 5

Note that each input line is required to terminate with a ‘#’ character as the end-of-the-line symbol.

The blank type of 6 for spaces can be left out when printing character types.

**Approaches:**

A rough structure of the program is as follows:

while (1) {

getline(inBuf);

int i = 0;

while (inBuf[i] != ‘#’) {

chType = search(inBuf[i]);

outBuf[i] = char(chType);

i++;

}

print outBuf;

clear inBuf;

clear outBuf;

}

1. Reading input lines

Two buffers need to be declared to save characters in an input line and to store output character types.

.data

inBuf: .space 80 # input line

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

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

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

.text

getline:

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

li $v0, 4 # Command code to write to the screen

syscall

la $a0, inBuf # Store the user input in inBuf

li $a1, 80 # inBuf has space for 80 character

li $v0, 8 # Command code to read a new line

syscall

jr $ra # Return

2. Character Search

A simple approach to finding a character type is to arrange all characters and their types into a 2D array. This makes future changes and updates to the character set a quick process. Use the MIPS table tabChar below and write either a linear or a binary search function to search for a character. Be careful that a character from the input string is stored as one byte, whereas the search table below is organized in units of words (4-bytes each). 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 .asm MIPS source code as your last name and submit the .asm file.

**tabChar table:**

.data

tabChar:

.word 0x09, 6 # tab

.word 0x0a, 6 # line feed

.word 0x0d, 6 # carraige return

.word ' ', 6

.word ‘#’, 5

.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 # ‘\’ used as the end-of-table symbol