Ean Dodge

```
1.  # Starter code for reversing the case of a 30 character input.
2.  # Put in comments your name and date please.  You will be
3.  # revising this code.
4.  #
5.  # Created by Dianne Foreback
6.  # Students should modify this code
7.  # Ean Dodge
8.  # 11/16/2023
9.  #
10. # This code displays the authors name (you must change
11. # outpAuth to display YourFirstName and YourLastName".
12. #
13. # The code then prompts the user for input
14. # stores the user input into memory "varStr"
15. # then displays the users input that is stored in"varStr"
16. #
17. # You will need to write code per the specs for
18. # procedures main, revCase and function findMin.
19. #
20. # revCase will to reverse the case of the characters
21. # in varStr.  You must use a loop to do this.  Another buffer
22. # varStrRev is created to hold the reversed case string.
23. #
24. # Please refer to the specs for this project, this is just starter code.
25. #
26. # In MARS, make certain in "Settings" to check
27. # "popup dialog for input syscalls 5,6,7,8,12"
28. #
29.         .data     # data segment
30.    .align 2   # align the next string on a word boundary
31. outpAuth:   .asciiz  "This is Ean Dodge presenting revCaseMin.\n"
32. outpPrompt: .asciiz  "Please enter 30 characters (upper/lower case mixed):\n"
33.    .align 2   #align next prompt on a word boundary
34. outpStr:    .asciiz  "You entered the string: "
35.         .align 2   # align users input on a word boundary
36. varStr:    .space 32  # will hold the user's input string thestring of 20 bytes
37.                 # last two chars are \n\0  (a new line and null char)
38.                 # If user enters 31 characters then clicks "enter" or hits the
39.                 # enter key, the \n will not be inserted into the 21st element
40.                 # (the actual users character is placed in 31st element).  the
41.                 # 32nd element will hold the \0 character.
```

```
                # .byte 32 will also work instead of .space 32
        .align 2   # align next prompt on word boundary


outpStrRev: .asciiz   "Your string in reverse case is: "
        .align 2   # align the output on word boundary
varStrRev:  .space 32  # reserve 32 characters for the reverse case string
    .align 2   # align  on a word boundary
outpStrMin: .asciiz    "\nThe min ASCII character after reversal is: "
#
minChar :    .space 10

        .text      # code section begins
        .globl       main
main:
#
# system call to display the author of this code
#
 la $a0,outpAuth          # system call 4 for print string needs address of string in $a0
 li $v0,4            # system call 4 for print string needs 4 in $v0
 syscall

#
# system call to prompt user for input
#
 la $a0,outpPrompt        # system call 4 for print string needs address of string in $a0
 li $v0,4            # system call 4 for print string needs 4 in $v0
 syscall
#
# system call to store user input into string thestring
#
li $v0,8            # system call 8 for read string needs its call number 8 in $v0
                        # get return values
la $a0,varStr      # put the address of thestring buffer in $t0
li $a1,32               # maximum length of string to load, null char always at end
# but note, the \n is also included providing total len < 22
       syscall
       #move $t0,$v0                 # save string to address in $t0; i.e. into "thestring"
#
# system call to display "You entered the string: "
#
 la $a0,outpStr  # system call 4 for print string needs address of string in $a0
 li $v0,4            # system call 4 for print string needs 4 in $v0
 syscall
```

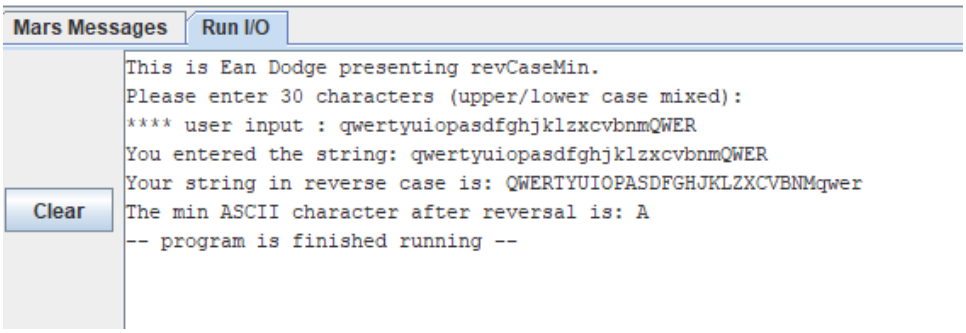
```
#
# system call to display user input that is saved in "varStr" buffer
#
 la $a0,varStr     # system call 4 for print string needs address of string in $a0
 li $v0,4          # system call 4 for print string needs 4 in $v0
 syscall
#
# Your code to invoke revCase goes next
#

la $a0 varStr              # put input into the argument
li $a1 30                  # put 30 into argument
jal revCase                # go to function


        # Exit gracefully from main()
              li   $v0, 10      # system call for exit
              syscall          # close file


        ######################################################################
        # revCase() procedure can go next
        ######################################################################
        #  Write code to reverse the case of the string.  The base address of the
        # string should be in $a0 and placed there by main().  main() should also place into
        # $a1 the number of characters in the string.
        #  You will want to have a label that main() will use in its jal
        # instruction to invoke revCase, perhaps revCase:
        #
        revCase:

        la $t1 varStrRev
        li $t0 0                              #used as increment
        li $t1 0                              #used for address of element
        li $t3 96                                     #used to figure if lower case

        reverse_loop:
        add      $t1, $a0, $t0     # $a0 is the base address for our 'input' array, add loop index
        lb       $t4, 0($t1)       # load a byte at a time according to counter
        beqz     $t4, exit                 # We found the null-byte
        beq    $t4, 10, exit               # we found the endline
        bge    $t3, $t4, toLower      # if it is upper case, need to add 32
        subiu   $t4, $t4, 32          # if lower case, need to subtact 32
        j reverse_exit
```

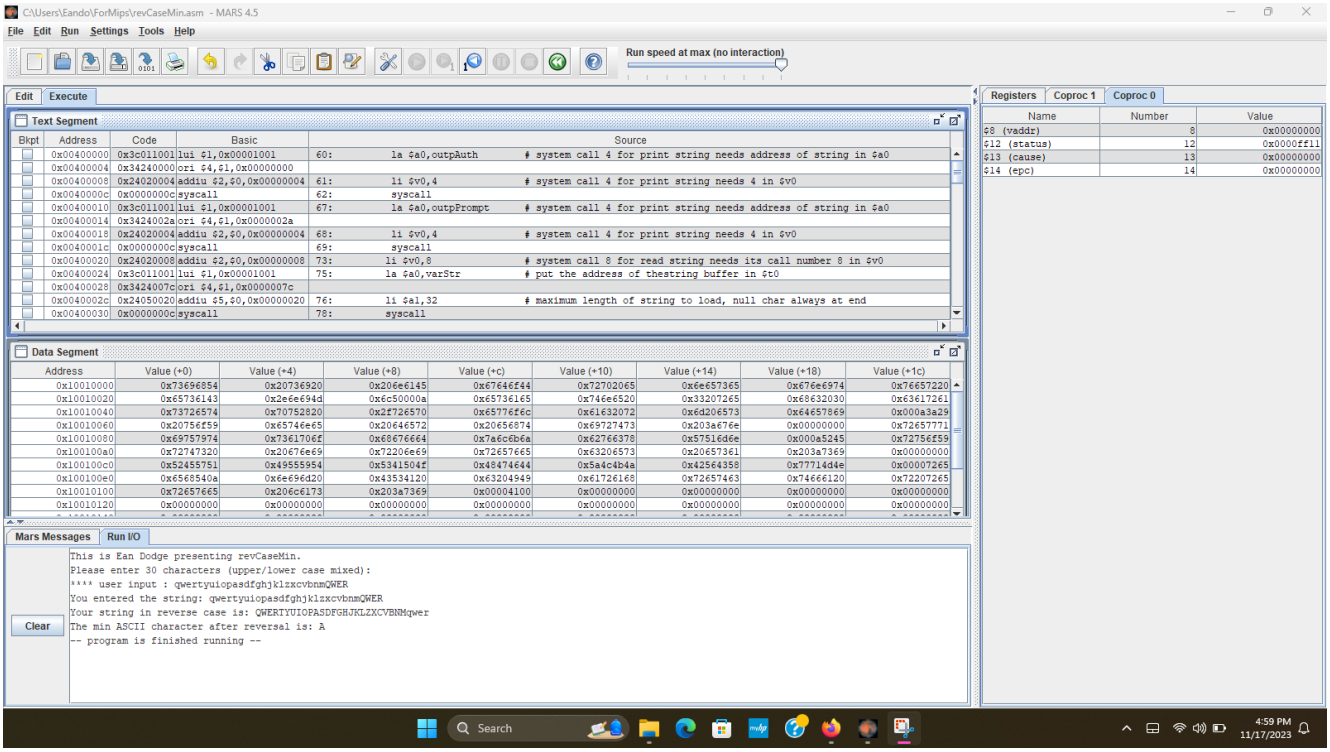
```
toLower:
addiu   $t4, $t4, 32          # add 32 to make lower case
reverse_exit:

sb        $t4, varStrRev($t0)# Overwrite this byte address in memory


addi      $t0, $t0, 1                 # Advance our counter (i++)
j         reverse_loop                # Loop until we reach our condition

exit:
move $a0, $t5                         #put $a0 in $t5 for later
move $a1, $t6



#
# After reversing the string, you may print it with the following code.
# This is the system call to display "Your string in reverse case is: "
 la $a0,outpStrRev # system call 4 for print string needs address of string in $a0
 li $v0,4            # system call 4 for print string needs 4 in $v0
 syscall
# system call to display the user input that is in reverse case saved in the varRevStr
 la $a0,varStrRev  # system call 4 for print string needs address of string in $a0
 li $v0,4            # system call 4 for print string needs 4 in $v0
 syscall

move $t5, $a0              # put back into arguments
move $t6, $a1
#
# Your code to invoke findMin() can go next
j findMin                     # go to find min function
# Your code to return to the caller main() can go next



#####################################################################
# findMin() function can go next
#####################################################################
#  Write code to find the minimum character in the string.  The base address of the
# string should be in $a0 and placed there by revCase.  revCase() should also place into
# $a1 the number of characters in the string.
#  You will want to have a label that revCase() will use in its jal
# instruction to invoke revCase, perhaps findMin:
```

```
174.        #
175.        #
176.        findMin:
177.        # write use a loop and find the minimum character
178.        li $t2 122                  #top value
179.        find_min_loop:
180.
181.        lb $t1,($a0)                    #get value in array
182.        beq $t1,$zero,print_out         #if its zero, then leave
183.
184.        addiu $a0,$a0,1                 #increment array
185.        blt $t2,$t1,find_min_loop       #If $t2 is less than, than start over
186.        move $t2,$t1                    #if not, move it into $t1
187.
188.        j find_min_loop             #start loop again until zero is found
189.
190.        print_out:
191.        sb $t2, minChar             # put answer into min Char
192.
193.
194.        #
195.        # system call to display "The min ASCII character after reversal is:  "
196.         la $a0,outpStrMin # system call 4 for print string needs address of string in $a0
197.         li $v0,4            # system call 4 for print string needs 4 in $v0
198.         syscall
199.
200.        # write code for the system call to print the the minimum character
201.        la $a0, minChar # system call 4 for print string needs address of string in $a0
202.        li $v0, 4           # system call 4 for print string needs 4 in $v0
203.        syscall
204.
205.
206.        # write code to return to the caller revCase() can go next
207.             jr $ra                         #go back to main
```

Brief description:

To do the reverse case I made a loop to go through the charcter array by incrementing through each element. In each element, I figure if it is lower case or upper case, or a zero (or endline). If it is lower case then I subtract 32, as this is traversing the asciiz table. If it is upper case, then I add 32. If it is a zero, or endline, then I skip and go to findMIn. Once I call the findMin function, it will look through each element, and figure if it is zero or not. If it is zero, it will skip through. If not a zero, then it will find the smallest number by looking at the decimal value. Once it finds the smallest value, it will put it into minChar, and use that to print.





Conclusion:

I learned how to use the jal and jr and what they do. I had trouble to find an algorithm to get upper and lower case, but got help from tutors