**FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)**

****

**COMPUTER ORGANIZATION AND ASSEMBLY COMP300B**

**SPRING 2021**

**FINAL EXAM 1 (week 15)**

**MIPS Programming with Strings**

**TAHA HASSAN (20-10643)**

**AHMAD SAAD (231461463)**

**MUHAMMAD SAMEED GILLANI (231488347)**

**You should place the graded (or ungraded in case of assignment) handout as second page of this report.**

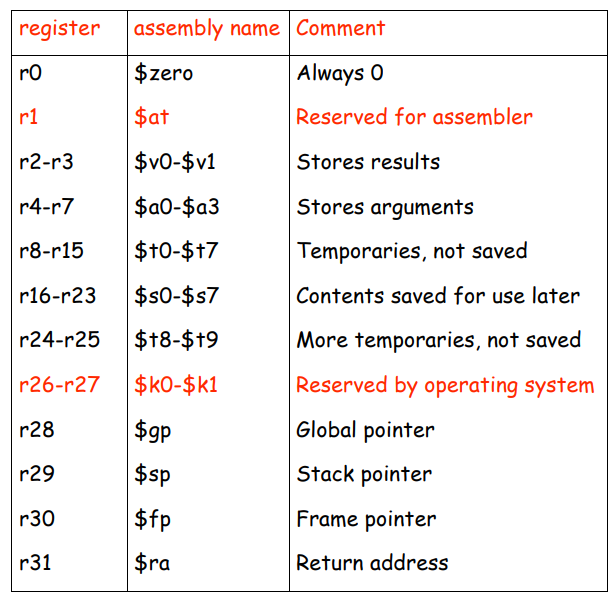
**From third page onwards following headings should be included:**

* **Introduction**
  + **Should carry information of all major library functions.**
* **Your logic / algorithm in simple English. Bullet points are appreciated.**
* **Your code**
* **Screen shots of at least three outputs of your code with appropriate inputs.**
* **References**

**Introduction**

In this 1st part of the final exam we had to write code in mips assembly language for 6 different programs, In the 1st task we had to write a program that should get a string from the user as input and the program calculates the length of the string, displays it as a result. For the 2nd task we had to write a program which again takes a string as user input and then converts the entire string to the upper case letters, the opposite of this is done in the 3rd task where we provide a string as input and the program converts that string to lowercase and displays the result. For the 4th task we ask a user to input any 2 strings and one integer value n. It then compares first n characters of both the string and displays the output. Output is 1 if the n characters from both string match. In the 5th task, we had to write a program that can find the no. of occurrences of a character in a string, the string is taken as a user input. For the 6th and final task we had to write a find and replace character, program in mips assembly language, in this task a user would be prompted to enter a string, after that he would be prompted to enter the character we need to replace from the previously entered string, then by which character we want to replace that string with, once everything is provided to the program it finds all the occurrences of the character that we want to replace, and then replace it with the character we entered to replace it with, the program then prints the new modified string, else if a character is entered and no occurrence of that character is found in the string, then our program should print that the following character was not found in the string for example “Could not found character s in the string” etc.

**Registers used:**



.data Introduces the data (i.e. global variables) section of the program.

.text Introduces the text (i.e. code) section of the program.

**Logic/ Algorithm**

**Task 1:**

* First, we need to define some global variables for our prompt and input so, we define 2 variables, the first is a prompt for “Enter String:” and the second one is a prompt for the output “The length of the string is: ”.
* Then, in the .text part we start to write our code, first we get the prompt printed to ask for a user input for string, then we get the string as input from the user and store it.
* Now, we set the counter to -1, and get the string into a temporary register $t1.
* After that we write a function for callength which calculates the length of the string, by first checking for the null character, then incrementing the string pointer and then the counter, and then we return to the top of loop.
* At last we print the calculated length of the string and gracefully exit the program.

**Task 2:**

* In the data section we create the appropriate prompts and assign space for the users string input.
* In the text section, we first print a prompt asking the user to input a string. Then we read that string and store.
* We then load the users input into $s0 and initialize $t0, with 0. To act as an index.
* We move onto ConvertUpperCase which will loop until we reach the end of the string.
* We add $s0 and $t0 to $s1. This will act like we are iterating through the user’s input. According to the value of $t0, we skip that many letters from the start of the input.  
  For example, if $t0 = 2 and the users input string is, ‘abcd’. Then $s1 will have, cd stored in it. In a way skipping the first 2 letters of the input string.
* We move on to load the first letter from $s1 into $t3.
* The exit condition is when $t3 = 0. i.e., the end of the string
* Then there is a block of code, which compares each lower-case alphabet with the letter loaded into $t3. When it matches the letter, it branches to that letter’s appropriate label.  
  Where that lowercase letter is replaced with its upper-case version in the string. And we jump back to the start of ConvertUpperCase.
* Since the letter is already upper case, we go on and increment $t0 by 1. And we loop over, ConvertUpperCase.
* Now with $t0 incremented, the next letter of the input string is loaded, and the same process is performed on it.
* When there is a non-alpha character, it is skipped and we only increment $t0.
* Once we reach the end of the string, we exit the loop and branch to, Print String.
* Here we print some text, saying that the following is the upper-case version of the original input.
* We then print the upper-case version of the input string and jump to, Exit.
* Here we simple terminate the program gracefully.

**Task 3:**

* Task 3 performs in the exact same way as Task2. But instead converts all lowercase letters to upper case.
* In the data section we create the appropriate prompts and assign space for the user’s string input.
* In the text section, we first print a prompt asking the user to input a string. Then we read that string and store.
* We then load the users input into $s0 and initialize $t0, with 0. To act as an index.
* We move onto ConvertLowerCase which will loop until we reach the end of the string.
* We add $s0 and $t0 to $s1. This will act like we are iterating through the user’s input. According to the value of $t0, we skip that many letters from the start of the input.  
  For example, if $t0 = 2 and the users input string is, ‘abcd’. Then $s1 will have, cd stored in it. In a way skipping the first 2 letters of the input string.
* We move on to load the first letter from $s1 into $t3.
* The exit condition is when $t3 = 0. i.e., the end of the string
* Then there is a block of code, which compares each upper-case alphabet with the letter loaded into $t3. When it matches the letter, it branches to that letter’s appropriate label.  
  Where that uppercase letter is replaced with its lower-case version in the string. And we jump back to the start of ConvertLowerCase.
* Since the letter is already lower case, we go on and increment $t0 by 1. And we loop over, ConvertLowerCase.
* Now with $t0 incremented, the next letter of the input string is loaded, and the same process is performed on it.
* When there is a non-alpha character, it is skipped and we only increment $t0.
* Once we reach the end of the string, we exit the loop and branch to, PrintString.
* Here we print some text, saying that the following is the lower-case version of the original input.
* We then print the lower-case version of the input string and jump to, Exit.
* Here we simple terminate the program gracefully.

**TASK 4:**

* In data section we declare 4 global variables for the prompts.
* In the section we start to write our code, and one by one prints each prompt for a user input.
* We used stack pointers to iterate through each string index by index.
* We incremented a count variable for every index we compared.
* If the variable reaches the number of characters the user told the program to compare, the loop breaks with a positive value.
* On the contrary, if two characters do not match, the loop breaks with a 0.

**TASK 5:**

* For question 5, we used a similar technique but without stack pointers.
* We ask user for two inputs, a string and a character.
* We iterate through the string, index by index and if the character at any index is the same as the character the user gave me.
* Then we increment a separate count variable, which I ultimately print at the end.
* And finally, gracefully exit the program.

**TASK 6:**

* First, we declare the prompts we need for this program in the .data section.
* Then we print the enter string prompt to get the user input.
* Now we again print a prompt to get the character to replace input, after reading it we immediately store the byte of character using $sb and move the value of the character to register $s2 for later use.
* Then we again print a prompt to get the character to replace the character in the string with, and then move the value to that character to another register.
* Then we check for null character and if found we go to label 1.
* And if the string doesn’t match, we go to label 2.
* In label 2 the string address is incremented.
* And then we move back to the top or label 3.
* But if the string matches then we replace it with the new character.
* If a null character is read, we go to label 1 where, the output is finally displayed on the screen as result for the find n replace.
* And if no character match is found it goes to label 4, where we get the character that was given as input by the user and output that the following character wasn’t found it the string.
* Finally in label 5 we gracefully exit the program.

**CODE**

**TASK 1:**

# Program Name: Task 1.asm

# Group Number: 5

# Group Members:

# Taha Hassan, 20-10643

# Muhammad Sameed Gilani, 231488347

# Ahmad Saad, 231461463

##############################

.data

prompt: .asciiz"Enter String: "

inputstring: .space 1024

outputmsg: .asciiz "Length of string is: "

.text

#printing prompt

li $v0,4

la $a0,prompt

syscall

#getting the user input for the string

li $v0, 8

la $a0, inputstring

li $a1, 1024

syscall

#setting the counter

li $t0, -1

#getting the string into a temporary register

la $t1, inputstring

#calculating the length of the string

callength:

lb $t2, 0($t1)

beqz $t2, label #check for null character

add $t1, $t1, 1 #incrementing the string pointer

add $t0, $t0, 1 #incrementing the counter

j callength #back to the top of loop

label:

#displaying the output message

li $v0,4

la $a0, outputmsg

syscall

#final result

li $v0, 1

move $a0, $t0

syscall

#graceful exit

li $v0,10

syscall

**TASK 2:**

# Program Name: Task 2.asm

# Group Number: 5

# Group Members:

# Taha Hassan, 20-10643

# Muhammad Sameed Gilani, 231488347

# Ahmad Saad, 231461463

##############################

.data

enterString: .asciiz "Enter string: \n"

text: .space 101

text\_size: .word 100

upperString: .asciiz "Upper case version of the string is: \n"

.text

main:

# Print a prompt, asking user to enter a string

li $v0,4

la $a0,enterString

syscall

# Create space in memory then read and writes the users input string into memory.

li $v0,8

la $a0,text

lw $a1,text\_size

syscall

la $s0,text # Loading the users input string into, $s0

li $t0,0 # $t0 will act as an index/counter, that increments each loop,

## that we will use to access each letter in the string.

ConvertUpperCase:

add $s1, $s0, $t0 # According to the value of $t0, we "skip" n letters from the start.

## If $t0 = 2 then $s1 has the string starting from the 3rd letter

### Ex: if $t0 = 2 and $s0 = abcd. Then $s1 = cd

lb $t3,($s1) # Loads byte, i.e. the first letter from $s1

## Ex: If $s1 = cd. Then $t3 = c

beqz $t3,PrintString # Exit condition. When the letter, i.e byte loaded in $t3 is = 0. We exit, as we have

## reached the end of the string.

# Checks each letter of the input string, and then branches to the corresponding upper case letter conversion lable.

# If the letter is already Upper Case or is non-alpha, then it is skipped and $t0 is incremented and ConvertUpperCase loops over.

beq $t3,'a',A

beq $t3,'b',B

beq $t3,'c',C

beq $t3,'d',D

beq $t3,'e',E

beq $t3,'f',F

beq $t3,'g',G

beq $t3,'h',H

beq $t3,'i',I

beq $t3,'j',J

beq $t3,'k',K

beq $t3,'l',L

beq $t3,'m',M

beq $t3,'n',N

beq $t3,'o',O

beq $t3,'p',P

beq $t3,'q',Q

beq $t3,'r',R

beq $t3,'s',S

beq $t3,'t',T

beq $t3,'u',U

beq $t3,'v',V

beq $t3,'w',W

beq $t3,'x',X

beq $t3,'y',Y

beq $t3,'z',Z

addi $t0,$t0,1 # Increments $t0 ( the index/counter ) to access the next letter in the string.

j ConvertUpperCase # Loops back to the start

PrintString:

# Prints text, that the following is the Upper Case output

li $v0,4

la $a0,upperString

syscall

# Prints the users input, in Upper Case

li $v0,4

move $a0,$s0

syscall

jal Exit

Exit:

# Exits the program gracefully

li $v0,10

syscall

# The following simply overwrites the Upper Case version of the letter at the position that

## the lower case version of it existed, in the input string

A:

li $t3,'A' # Loads the Upper Case letter into $t3

sb $t3,($s1) # stores the byte (Overwrites the letter) at its original position in the input string.

j ConvertUpperCase # Jumps back to ConvertUpperCase to process the remaining letters in the input string.

# This same process happens for each of the following alphabets.

B:

li $t3,'B'

sb $t3,($s1)

j ConvertUpperCase

C:

li $t3,'C'

sb $t3,($s1)

j ConvertUpperCase

D:

li $t3,'D'

sb $t3,($s1)

j ConvertUpperCase

E:

li $t3,'E'

sb $t3,($s1)

j ConvertUpperCase

F:

li $t3,'F'

sb $t3,($s1)

j ConvertUpperCase

G:

li $t3,'G'

sb $t3,($s1)

j ConvertUpperCase

H:

li $t3,'H'

sb $t3,($s1)

j ConvertUpperCase

I:

li $t3,'I'

sb $t3,($s1)

j ConvertUpperCase

J:

li $t3,'J'

sb $t3,($s1)

j ConvertUpperCase

K:

li $t3,'K'

sb $t3,($s1)

j ConvertUpperCase

L:

li $t3,'L'

sb $t3,($s1)

j ConvertUpperCase

M:

li $t3,'M'

sb $t3,($s1)

j ConvertUpperCase

N:

li $t3,'N'

sb $t3,($s1)

j ConvertUpperCase

O:

li $t3,'O'

sb $t3,($s1)

j ConvertUpperCase

P:

li $t3,'P'

sb $t3,($s1)

j ConvertUpperCase

Q:

li $t3,'Q'

sb $t3,($s1)

j ConvertUpperCase

R:

li $t3,'R'

sb $t3,($s1)

j ConvertUpperCase

S:

li $t3,'S'

sb $t3,($s1)

j ConvertUpperCase

T:

li $t3,'T'

sb $t3,($s1)

j ConvertUpperCase

U:

li $t3,'U'

sb $t3,($s1)

j ConvertUpperCase

V:

li $t3,'V'

sb $t3,($s1)

j ConvertUpperCase

W:

li $t3,'W'

sb $t3,($s1)

j ConvertUpperCase

X:

li $t3,'X'

sb $t3,($s1)

j ConvertUpperCase

Y:

li $t3,'Y'

sb $t3,($s1)

j ConvertUpperCase

Z:

li $t3,'Z'

sb $t3,($s1)

j ConvertUpperCase

**TASK 3:**

# Program Name: Task 3.asm

# Group Number: 5

# Group Members:

# Taha Hassan, 20-10643

# Muhammad Sameed Gilani, 231488347

# Ahmad Saad, 231461463

##############################

.data

enterString: .asciiz "Enter string: \n"

text: .space 101

text\_size: .word 100

lowerString: .asciiz "Lower case version of the string is: \n"

.text

main:

# Print a prompt, asking user to enter a string

li $v0,4

la $a0,enterString

syscall

# Create space in memory then read and writes the users input string into memory.

li $v0,8

la $a0,text

lw $a1,text\_size

syscall

la $s0,text # Loading the users input string into, $s0

li $t0,0 # $t0 will act as an index/counter, that increments each loop,

## that we will use to access each letter in the string.

ConvertLowerCase:

add $s1, $s0, $t0 # According to the value of $t0, we "skip" n letters from the start.

## If $t0 = 2 then $s1 has the string starting from the 3rd letter

### Ex: if $t0 = 2 and $s0 = abcd. Then $s1 = cd

lb $t3,($s1) # Loads byte, i.e. the first letter from $s1

## Ex: If $s1 = cd. Then $t3 = c

beqz $t3,PrintString # Exit condition. When the letter, i.e byte loaded in $t3 is = 0. We exit, as we have

## reached the end of the string.

# Checks each letter of the input string, and then branches to the corresponding lower case letter conversion lable.

# If the letter is already lower Case or is non-alpha, then it is skipped and $t0 is incremented and ConvertLowerCase loops over.

beq $t3,'A',a

beq $t3,'B',bb

beq $t3,'C',c

beq $t3,'D',d

beq $t3,'E',e

beq $t3,'F',f

beq $t3,'G',g

beq $t3,'H',h

beq $t3,'I',i

beq $t3,'J',jj

beq $t3,'K',k

beq $t3,'L',l

beq $t3,'M',m

beq $t3,'N',n

beq $t3,'O',o

beq $t3,'P',p

beq $t3,'Q',q

beq $t3,'R',r

beq $t3,'S',s

beq $t3,'T',t

beq $t3,'U',u

beq $t3,'V',v

beq $t3,'W',w

beq $t3,'X',x

beq $t3,'Y',y

beq $t3,'Z',z

addi $t0,$t0,1 # Increments $t0 ( the index/counter ) to access the next letter in the string.

j ConvertLowerCase # Loops back to the start

PrintString:

# Prints text, that the following is the lower Case output

li $v0,4

la $a0,lowerString

syscall

# Prints the users input, in lower Case

li $v0,4

move $a0,$s0

syscall

jal Exit

Exit:

# Exits the program gracefully

li $v0,10

syscall

# The following simply overwrites the Lower Case version of the letter at the position that

## the Upper case version of it existed, in the input string

a:

li $t3,'a' # Loads the lower Case letter into $t3

sb $t3,($s1) # stores the byte (Overwrites the letter) at its original position in the input string.

j ConvertLowerCase # Jumps back to ConvertLowerCase to process the remaining letters in the input string.

# This same process happens for each of the following alphabets.

bb:

li $t3,'b'

sb $t3,($s1)

j ConvertLowerCase

c:

li $t3,'c'

sb $t3,($s1)

j ConvertLowerCase

d:

li $t3,'d'

sb $t3,($s1)

j ConvertLowerCase

e:

li $t3,'e'

sb $t3,($s1)

j ConvertLowerCase

f:

li $t3,'f'

sb $t3,($s1)

j ConvertLowerCase

g:

li $t3,'g'

sb $t3,($s1)

j ConvertLowerCase

h:

li $t3,'h'

sb $t3,($s1)

j ConvertLowerCase

i:

li $t3,'i'

sb $t3,($s1)

j ConvertLowerCase

jj:

li $t3,'j'

sb $t3,($s1)

j ConvertLowerCase

k:

li $t3,'k'

sb $t3,($s1)

j ConvertLowerCase

l:

li $t3,'l'

sb $t3,($s1)

j ConvertLowerCase

m:

li $t3,'m'

sb $t3,($s1)

j ConvertLowerCase

n:

li $t3,'n'

sb $t3,($s1)

j ConvertLowerCase

o:

li $t3,'o'

sb $t3,($s1)

j ConvertLowerCase

p:

li $t3,'p'

sb $t3,($s1)

j ConvertLowerCase

q:

li $t3,'q'

sb $t3,($s1)

j ConvertLowerCase

r:

li $t3,'r'

sb $t3,($s1)

j ConvertLowerCase

s:

li $t3,'s'

sb $t3,($s1)

j ConvertLowerCase

t:

li $t3,'t'

sb $t3,($s1)

j ConvertLowerCase

u:

li $t3,'u'

sb $t3,($s1)

j ConvertLowerCase

v:

li $t3,'v'

sb $t3,($s1)

j ConvertLowerCase

w:

li $t3,'w'

sb $t3,($s1)

j ConvertLowerCase

x:

li $t3,'x'

sb $t3,($s1)

j ConvertLowerCase

y:

li $t3,'y'

sb $t3,($s1)

j ConvertLowerCase

z:

li $t3,'z'

sb $t3,($s1)

j ConvertLowerCase

**TASK 4:**

# Program Name: Task 4.asm

# Group Number: 5

# Group Members:

# Taha Hassan, 20-10643

# Muhammad Sameed Gilani, 231488347

# Ahmad Saad, 231461463

##############################

.data

firstPrompt: .asciiz "Enter first string: "

secondPrompt: .asciiz "Enter second string: "

thirdPrompt: .asciiz "Enter number of characters to match: "

output: .asciiz "Output is: "

firstStr: .space 100

secondStr: .space 100

.text

main:

la $a0, firstPrompt # Print first string prompt for user

li $v0, 4

syscall

la $a0, firstStr # Get first string and store it in firstStr static data from user

li $a1, 100

li $v0, 8

syscall

la $a0, secondPrompt # Print second string prompt for user

li $v0, 4

syscall

la $a0, secondStr # Get second string and store it in secondStr static data from user

li $a1, 100

li $v0, 8

syscall

la $a0, thirdPrompt # Prompt to get user to integer value

li $v0, 4

syscall

li $v0, 5 # Get an integer value from user

syscall

la $a0, firstStr # Address for first string in $a0

la $a1, secondStr # Address for second string in $a1

move $a2, $v0 # Integer value n in $a2

jal strncmp

move $t0, $v0 # Temporarily store return value

la $a0, output # Print output message

li $v0, 4

syscall

move $a0, $t0 # Print the output

li $v0, 1

syscall

li $v0, 10 # Exit the program

syscall

strncmp:

addi $sp, $sp, -16 # Push the stack

sw $ra, 0($sp) # Store return address and arguments

sw $a0, 4($sp)

sw $a1, 8($sp)

sw $a2, 12($sp)

li $s0, 0 # Counter for loop

loop: beq $s0, $a2, equal # If the counter reaches n, strings are equal for n characters

lb $t0, 0($a0) # Load a character from first string

lb $t1, 0($a1) # Load a character from second string

bne $t0, $t1, notequal # If characters are not equal exit loop

addi $a0, $a0, 1 # Increment the addesses

addi $a1, $a1, 1

add $s0, $s0, 1 # Increment the counter

j loop

equal: # Sets return value $v0 to 1 since the strings are equal

li $v0, 1

j done

notequal: # Sets return value $v0 to 0 since the strings are not equal.

li $v0, 0

done: # Exit the function.

lw $ra, 0($sp) # Retrieve return address

addi $sp, $sp, 16 # Pop the stack

jr $ra

**TASK 5:**

# Program Name: Task 5.asm

# Group Number: 5

# Group Members:

# Taha Hassan, 20-10643

# Muhammad Sameed Gilani, 231488347

# Ahmad Saad, 231461463

##############################

.data

prompt: .asciiz "Enter String: "

promptTwo: .asciiz "Enter character: "

nextLine: .asciiz "\n"

occurances: .asciiz "Frequency of occurrence of given character: "

string: .space 100 # Maximum string size the program accepts is 100

.text

main:

li $v0, 4 # Asking user for input

la $a0, prompt

syscall

la $a0, string # Here the string is being read and stored

li $a1, 100

li $v0, 8

syscall

move $a3, $a0 # Here I move the string to a different register

li $v0, 4 # I also ask the user for a character input

la $a0, promptTwo

syscall

li $v0, 12 # Here I read and store the character

syscall

move $s2, $v0 # I store the character in register $s2

li $s1, 100 # $s1 will be upper bound for my loop as it is the string length

li $t4, 0 # This is increment variable for character repitition count

li $t0, 0 # This is the index being observed

bge $t0, $s1, finalPrint # This is the break condition

increment:

lb $s0,($a3) # Register $s0 reads every index of the string one by one

bne $s0, $s2, noIncrement # If the character and the index do not match, there is no increment

add $t4, $t4, 1 # If they do match, 1 is added to the increment variable

noIncrement:

addiu $a3, $a3, 1 # Index for the string is incremented here

beq $s0, 0, finalPrint # finalPrint function is called if the string has ended

j increment # Otherwise, increment function is recalled

finalPrint:

li $v0, 4 # Moving to next line

la $a0, nextLine

syscall

li $v0, 4 # Printing appropriate output prompt

la $a0, occurances

syscall

li $v0, 1 # Printing the amount of occurances of character as an integer

move $a0, $t4 # Moving the count from $t4 to $a0

syscall

li $v0, 10 # Graceful exit

syscall

**TASK 6:**

# Program Name: Task 6.asm

# Group Number: 5

# Group Members:

# Taha Hassan, 20-10643

# Muhammad Sameed Gilani, 231488347

# Ahmad Saad, 231461463

##############################

.data

prompt1: .asciiz"Enter String: "

prompt2: .asciiz"Enter character to find: "

prompt3: .asciiz"\nEnter character to replace: "

prompt4: .asciiz"\nThe find-n-replace results in: "

prompt5: .asciiz"\nCould not find character "

prompt6: .asciiz" in the string."

string: .space 100

char: .space 1

.text

main:

#printing the string to get input

li $v0,4

la $a0,prompt1

syscall

#reading string

la $a0,string

li $a1, 100

li $v0,8

syscall

#printing string to get character input

li $v0,4

la $a0,prompt2

syscall

li $v0, 12

syscall

sb $v0,char #storing chracter

move $s2,$v0 #moving it to $s2

#printing string to get character to replace with

li $v0,4

la $a0,prompt3

syscall

li $v0,12

syscall

move $t1,$v0 #moving content to $t1

li $t3,0

la $a0,string

label3:

lb $t2,0($a0) #$t2 a character from string

beqz $t2,label1

beq $t2, '\n', label1 #if $t2 null, or $t2=/n goto label1

bne $s2,$t2,label2 #if the string doesnt match go to label2

sb $t1,0($a0) #replacing character

li $t3,1 #set flag to 1

label2: #for incrementing string address

addi $a0,$a0,1

j label3

label1:

# Print a message with the replacement results

beqz $t3,label4

li $v0,4

la $a0, prompt4

syscall

li $v0,4

la $a0,string

syscall

j label5

#if the character is not found in the string

label4:

li $v0,4

la $a0, prompt5

syscall

li $v0,4

la $a0, char

syscall

li $v0,4

la $a0, prompt6

syscall

#gracefull exit

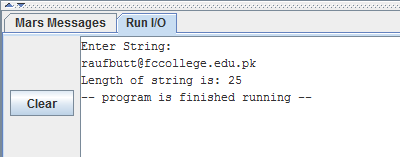
label5:

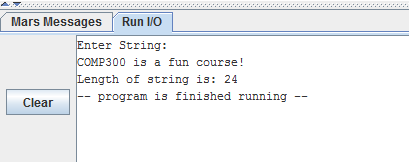
li $v0,10

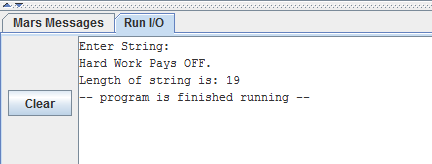
syscall

**SCREENSHOTS**

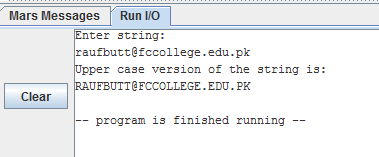
TASK 1

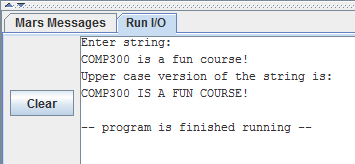


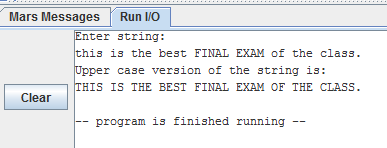




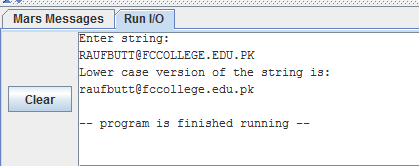
TASK 2

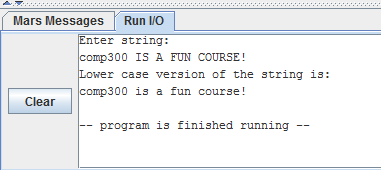


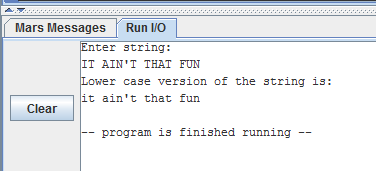




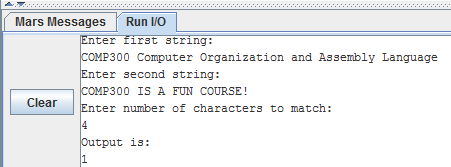
TASK 3

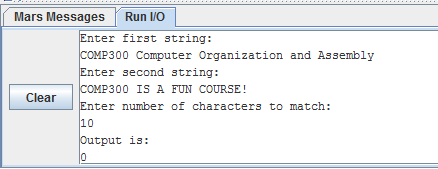


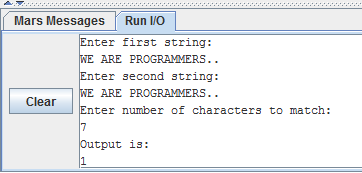




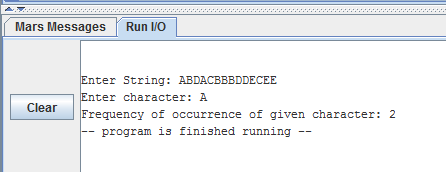
TASK 4

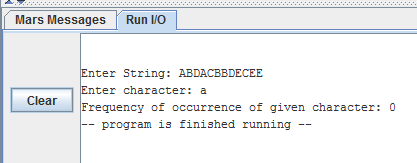


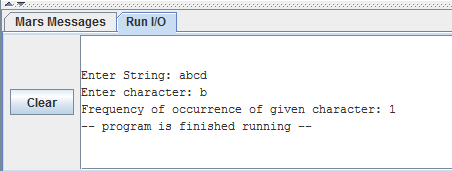




TASK 5







TASK 6

