.data

prompt: .asciiz "Enter a String: "

s: .space 100 # sting buffer length for user input

msg: .asciiz "New String: "

.text

.globl main

main:

#printing a prompt string onto the screen syscode is 4 for printing a string which will be loaded into $v0 and base address is in $a0

addi $v0,$0,4

la $a0,prompt

syscall

#syscode 8 for reading a string from user $a0 has base address and $a1 has a length

addi $v0, $zero, 8

la $a0, s

li $a1, 100

syscall

addi $t0,$0,0 #for old index

addi $t1,$0,0 #for new index

loop: lb $t2,s($t0) #loading a byte into $t2 register means char C

addi $t0,$t0,1 #incrementing old\_index

beq $t2,0x20,loop1 #checking if character is space or not ascii value of space is 32 (0x20) in hex if it is space branch to loop1

sb $t2,s($t1) #if not spcae storing the charqacter at new index value

addi $t1,$t1,1 #incrementing a new index

beqz $t2,done #if $t2 has null value jump to done

j loop # jump to loop

loop1:

j loop #jump to loop countinue statement

done: #printing a message

la $a0,msg

addi $v0,$0,4

syscall

#printing a string without space

la $a0,s

addi $v0,$0,4

syscall

#exit code

addi $v0,$0,10

syscall

—---------------------------------------------------------------------------------

.data

prompt: .asciiz "Enter a value of X(you want to find out the number of occurance): "

.align 1

array: .half 12 20 30 12 12 12 40 50 60 13

.align 0

N: .byte 10

F: .byte 0

.text

.globl main

main:

la $a1,array

lb $a2,N

la $a0,prompt

addi $v0,$0,4

syscall

addi $v0,$0,5

syscall

addi $a3,$v0,0

jal Occurrences

sb $v0,F

addi $a0,$v0,0

addi $v0,$0,1

syscall

addi $v0,$0,10

syscall

Occurrences:

addi $t0,$0,0

addi $v0,$0,0 #count

loop: beq $t0,$a2,done

lh $t1,0($a1)

addi $a1,$a1,2

addi $t0,$t0,1

beq $t1,$a3,inc\_count

j loop

inc\_count:

addi $v0,$v0,1

j loop

done:

jr $ra

—-------------------------------------------------------------------------------

.data

arr: .word 1, 2, 3, 4, 5

size: .word 5

.text

main:

la $s0, arr

lw $s1, size($0)

addi $s1, $s1, -1

addi $sp, $sp, -12

sw $s0, 0($sp)

sw $s1, 4($sp)

sw $ra, 8($sp)

jal reverse\_print

lw $s0, 0($sp)

lw $s1, 4($sp)

lw $ra, 8($sp)

addi $sp, $sp, 12

j exit

exit:

li $v0, 10

syscall

# recursive function to print array elements in reverse order

reverse\_print:

lw $t0, 0($sp)

lw $t1, 4($sp)

bltz $t1, base

sll $a0, $t1, 2

add $a0, $a0, $t0

lw $a0, 0($a0)

li $v0, 1

syscall

li $a0, 32

li $v0, 11

syscall

addi $sp, $sp, -12

sw $t0, 0($sp)

addi $t1, $t1, -1

sw $t1, 4($sp)

sw $ra, 8($sp)

jal reverse\_print

lw $ra, 8($sp)

addi $sp, $sp, 12

base:

jr $ra

**Q6**

.data

A: .word 1, 3, 5, 7, 9 #array

N: .word 5 #size of array

.text

.globl main

main:

la $s0, A #load address of array store in $s0

lw $s1, N($0) #load the size of array in $s1

addi $s1, $s1, -1 #incrementing size of array backwards

addi $sp, $sp, -12 #reserving space in the stack

sw $s0, 0($sp) #storing array in stack so its not altered by func

sw $s1, 4($sp) #storing the size of array in stack so its not altered by func

sw $ra, 8($sp) #storing the return address in stack so its not altered by func

jal reverse #call reverse func

lw $s0, 0($sp) #retrieving values from the stack

lw $s1, 4($sp)

lw $ra, 8($sp)

addi $sp, $sp, 12 #returning space back to the stack

j exit

exit: #exit and end program

li $v0, 10

syscall

reverse: # reverse func! decrement thru array and print char, space

lw $t0, 0($sp) #loading array from stack to temp $t0

lw $t1, 4($sp) #loading array size from stack to temp $t1

blt $t1, $zero, done #branch if array size less than 0

sll $a0, $t1, 2 #increment thru array

add $a0, $a0, $t0 #add the value of the array index into $a0

lw $a0, 0($a0) #print the value of the array at that index

li $v0, 1

syscall

li $a0, 32 #print character (space)

li $v0, 11

syscall

addi $sp, $sp, -12 #reserve space in stack

sw $t0, 0($sp) #store array in stack

addi $t1, $t1, -1 #decrement array size

sw $t1, 4($sp) #store array size in stack

sw $ra, 8($sp) #store return address in stack

jal reverse #go back to beginning of reverse func

lw $ra, 8($sp) #load return address from stack

addi $sp, $sp, 12 #giving space back to the stack

done: #return back to main

jr $ra