**CS224**

**Section No.: 02**

**Spring 2020**

**Lab No. 1**

**Defne Betül Çiftci/21802635**

**Solution for Part 1:**

.text

getTheSize:

#first ask the user size and then get the size

li $t2, 0

# la $t0, array #this is commented out because it gave some issues

#what i did instead was to store $s0 in the location array + $t0

#and increment $t0 by 4 each time

li $v0, 4

la $a0, getSize

syscall

li $v0, 5

syscall

move $s0, $v0

#check for everything and jump accordingly

bgt $s0, 20, ifGreaterThan

beq $s0, 20, main

blt $s0, 20, main

jr $ra

li $v0, 10

syscall

ifGreaterThan:

li $v0, 4

la $a0, enterSize

syscall

jal getTheSize

main:

beq $t2, 0, print #t2 is to control the jump between these two

li $v0, 5

syscall

move $s1, $v0

sb $s1, array($t0)

add $t3, $t3, $s1 #add each number

addi $t2, $t2, 4

addi $s0, $s0, -1 #decrement the size

bne $s0, 0, main #if size is not zero, run a loop. when the loop ends, it should

#go back to the "jal main" command in print

jr $ra

print:

addi $t2, $t2, 1

li $v0, 4

la $a0, arrayInit

syscall

jal main

li $v0, 4

la $a0, arrayEnd

syscall

li $v0, 4

la $a0, summation

syscall

li $v0, 1

move $a0, $t3

syscall

.data

getSize: .asciiz "Please enter the size you want : "

array: .space 80

enterSize: .asciiz "Enter the size you want again\n"

arrayInit: .asciiz "array: [ "

arrayEnd: .asciiz "]\n"

summation: .asciiz "Summation: "

**Solution for Part 2:**

.text

main:

li $v0, 4

la $a0, getAText

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, getBText

syscall

li $v0, 5

syscall

move $a1, $v0

li $v0, 4

la $a0, getCText

syscall

li $v0, 5

syscall

move $a2, $v0

jal getD

move $a0, $t0

jal theFunction

add $s0, $v0, $0

li $v0, 4

la $a0, printOutput

syscall

li $v0, 1

move $a0, $s0

syscall

li $v0, 10 #exits

syscall

getD:

li $v0, 4

la $a0, getDText

syscall

li $v0, 5

syscall

move $a3, $v0

blt $a3, 0, getD

beq $a3, 0, getD

jr $ra

addTilPositive:

add $a1, $a1, $a3

blt $a1, $a2, addTilPositive

theFunction: # x= a \* (b - c) % d

blt $a1, $a2, addTilPositive #add to a1 as the amount of a3 until a1-a2 is positive

sub $t0, $a1, $a2 # b-c = t0

blt $a0, 0, addTilMultiplyPos

addTilMultiplyPos: #add until the multiplier is positive

add $a0, $a0, $a3

blt $a0, 0, addTilMultiplyPos

mul $t1, $t0, $a0 # a \* (b-c) = t1

div $t1, $a3 #divides to a3

mfhi $s0 #remainder is stored in s0

add $v0, $s0, $0

jr $ra

.data

getDText: .asciiz "d= "

getAText: .asciiz "a= "

getCText: .asciiz "c= "

getBText: .asciiz "b= "

printOutput: .asciiz "The result of operation x= a \* (b - c) % d is : "