CS224
Section No.: 2
Spring 2019
Lab No.3
Zeynep Cankara/21703381

Answers for Question 1 and 2:

```
#*****************
#                               *
#       text segment *
#                               *
#*****************
        .text
        .globl __start
__start:
        ## ============  Question 1 ==================== ##
        # Prompt task
        la $a0, promptRecursiveDivision
        li $v0, 4
        syscall
        la $a0, promptDividend
        syscall
        # obtain the integer from the user
        li $v0, 5
        syscall
        move $s0, $v0
        # prompt the divisor
        la $a0, promptDivisor
        li $v0, 4
        syscall
        li $v0, 5
        syscall
        move $a1, $v0
        move $a0, $s0
        # Recursive function call
        jal recursiveDivision
        move $s0, $a0
        la $a0, resultQuotient
        li $v0, 4
        syscall
        move $a0, $s0
        li $v0, 1
        syscall # print quotient
        la $a0, resultRemainder
        li $v0, 4
        syscall
        move $a0, $a1
        li $v0, 1
        syscall # print remainder
```

```
        ## ===========  Question 2 ================== ##
        # Prompt Question 2 task
        la $a0, promptMultiplyDigits
        li $v0, 4
        syscall
        la $a0, promptNumber
        syscall
        # obtain the integer from the user
        li $v0, 5
        syscall
        move $a0, $v0
        # Function call
        jal multiplyDigits
        # store result s0
        move $s0, $v0
        la $a0, resultMultDigits
        li $v0, 4
        syscall
        move $a0, $s0
        li $v0, 1
        syscall

        li $v0, 10
        syscall # exit program

# Function divides two numbers returns the quotient in $a0 reg
# Assumption1: numbers are positive
recursiveDivision:
        # malloc
        subi $sp, $sp, 12
        sw   $a1, 8($sp)
   sw   $a0, 4($sp)
   sw   $ra, 0($sp)
   # base condition
   # if num1 < num2 else
   bgt $a0, $a1, elseDiv
   move $t0, $a0 # obtain the remainder, remainder = num1
   addi $a0, $0, 0        # return 0, quotient = 0
   move $a1, $t0
   addi $sp, $sp, 12  # dealloc
   jr  $ra   # goto next
elseDiv:
        # num1 = num1-num2
        sub $a0, $a0, $a1
        # recursive function call
        jal recursiveDivision # return 1 + recursiveDivision(num1-num2, num2)
   # load arguments back to reg
   lw   $ra, 0($sp)
   addi $sp, $sp, 12  # dealloc stack pointer
   addi $a0, $a0, 1 # quotient += 1
        jr $ra # goto next
```

```
## Function takes an integer returns multiplication of it's digits
# int multiplyDigits(int number ($a0))
multiplyDigits:
        # alloc stack space
        subi $sp, $sp, 8
        sw $a0, 4($sp)
        sw $ra, 0($sp)
        # base case
        bgt $a0, $0, elseMul # if (num <= 0)
        addi $v0, $0, 1# return 1
        addi $sp, $sp, 8  # dealloc
        jr  $ra   # goto next
elseMul:
        div $a0, $a0, 10
        mflo $a0 # obtain the quotient
        jal multiplyDigits # else {return (int)(num % 10) * multiplyDigit(num/10);}
        lw $a0, 4($sp)  # pop a0 from the stack
   div $t0, $a0, 10
   mfhi $t0
   mul $v0, $v0, $t0 # (int)(num % 10)
   # load arguments back to reg
   lw   $ra, 0($sp)
   addi $sp, $sp, 8  # dealloc stack pointer
        jr $ra # goto next


#******************
#                               *
#       data segment  *
#                               *
#******************
        .data
promptRecursiveDivision: .asciiz "Function divides 2 numbers returns the quotient and the
remainder..."
promptDividend: .asciiz "\nPlease enter the Dividend (dividend > 0): "
promptDivisor: .asciiz "\nPlease enter the Divisor (divisor > 0): "
resultQuotient: .asciiz "\nQuotient: "
resultRemainder: .asciiz "\nRemainder: "
promptMultiplyDigits: .asciiz "\nFunction multiplies digits of an integer recursively..."
promptNumber: .asciiz "\nPlease enter the number: "
resultMultDigits: .asciiz "\nMultiplication of digits: “
```

Answer for Question 3:

```
# @brief: Function deletes nodes with the value x.
# @params: $a0 head of the list. $a1 value nodes to be deleted contains
# @returns: $v0 nodes counted so far. $v1 pointer to head
### ANSWER: No because MIPS does not have an instruction for heap allocation.
# Thus, as a result of lack of memory allocation instructions memory leaks occur in MIPS.
```

```
Delete_x:
        # restore the new head pointer in $v1
        move $v1, $a0
        # count number of nodes in the linked list
        li $v0, 0
        beq $v1, $0, done
        # head node is not empty
        move $t0, $v1 # t0 = Node *prev = head;
        lw $t1, 0($t0) # t1 = Node *cur = head->next;
        # check whether head contains target or not
        checkHeadTarget:
                sne $t2, $0, $v1 # t2 = head != NULL
                lw $t3, 4($v1) # t3 = head->data
                seq $t4, $a1, $t3 # t4 = head->data == target
                and $t4, $t4, $t2 # (head != NULL && cur->data == target)
                bne $t4, 1, headNotTarget # head is not equals target
                move $v1, $t1  # head = head->next
                move $t0, $t1 # prev
                lw $t1, 0($t0) # cur
                addi $v0, $v0, 1 #  count++
                j checkHeadTarget
        headNotTarget:
                beq $t1, $0, done # cur != NULL
                lw $t3, 4($t1) # t3 = cur->data
                seq $t4, $a1, $t3 # t4 = cur->data == target
                beq $t4, 1, delNode
                lw $t1, 0($t1) # prev = prev->next;
                lw $t0, 0($t0) # cur = cur->next;
                # TODO: count++;
                j headNotTarget
                delNode:
                        lw $t4, 0($t1) # obtain next's pointer
                        sw $0, 0($t1) # make current pointer null
                        sw $t4, 0($t0) # prev->next = current->next
                        move $t1, $t4 # cur = cur->next;
                        addi $v0, $v0, 1 # count++;
                        j headNotTarget
        done:
         jr $ra #goto next
```