CS224
Section No.: 2
Spring 2019
Lab No. 3
Munib Emre Sevilgen / 21602416

Part 1:

#1.Question

```
recursiveDivision: #Recursive divide function
        addi $sp, $sp, -12
        sw $a0, 8($sp)
        sw $a1, 4($sp)
        sw $ra, 0($sp)

        bge $a0, $a1 else_recursiveDivision #Checks the current value is bigger than divider

        addi $v0, $zero, 0 #Set the quotient to zero
        add $v1, $zero, $a0 #Set the remainder

        lw $a1, 4($sp)
        lw $a0, 8($sp)
        addi $sp, $sp, 12
        jr $ra

else_recursiveDivision:
        sub $a0, $a0, $a1 #Update current value
        jal recursiveDivision
        addi $v0, $v0, 1 #Add one to quotient

        lw $ra, 0($sp)
        lw $a1, 4($sp)
        lw $a0, 8($sp)
        addi $sp, $sp, 12
        jr $ra
```

#2. Question

```
multiplyDigits: #Multiply digits function
        addi $sp, $sp, -12
        sw $a0, 8($sp)
        sw $s0, 4($sp)
```

```
        sw $ra, 0($sp)

        li $t0, 10 #Divider
        div $a0, $a0, $t0
        mfhi $s0 #Put the $a0 % 10 to $s0

        bgt $a0, 0 else_multiplyDigits #Checks whether the last digit or not

        add $v0, $zero, $s0 #Update return value with the first digit
        lw $s0, 4($sp)
        lw $a0, 8($sp)
        addi $sp, $sp, 12
        jr $ra

else_multiplyDigits:
        jal multiplyDigits
        mul $v0, $v0, $s0 #Update return value with the current digit

        lw $ra, 0($sp)
        lw $s0, 4($sp)
        lw $a0, 8($sp)
        addi $sp, $sp, 12
        jr $ra
```

#3.Question

```
Delete_x:
        addi $sp, $sp, -24
        sw $s0, 20 ($sp)
        sw $s1, 16 ($sp)
        sw $s2, 12 ($sp)
        sw $s3, 8 ($sp)
        sw $s4, 4 ($sp)
        sw $ra, 0 ($sp)

        move $s0, $a0 #The pointer to the current node
        move $s1, $zero #The pointer to the next node
        move $s2, $zero #The pointer to previous node
        move $s3, $s0 #The pointer to head
        addi $s4, $a1, 0 #The integer value of the element to be deleted

        li $t0, 0 #Counter to return
```

```
        checkFirst:
                beq $s0, $zero, end_Delete_x #If the head is NULL
                #Checks the first node
                lw $s1, 0($s0)
                lw $t1, 4($s0) #The integer value in current node
                beq $t1, $s4, deleteHead
                j loop

        deleteHead:
                #Delete the first node
                addi $t0, $t0, 1
                move $s3, $s1
                move $s0, $s1
                j checkFirst

        loop:
                beq $s1, 0, end_Delete_x

                move $s2, $s0 #Update previous node
                move $s0, $s1 #Update current node
                lw $s1, 0($s0) #Update next node

                checkNumber:
                #Check the current integer value
                lw $t1, 4($s0)
                beq $t1, $s4, deleteNode

                j loop

                deleteNode:
                        sw $s1, 0($s2) #Put the next node to previous node's next
                        move $s0, $s1 #Update the current node
                        addi $t0, $t0, 1 #Increase counter
                        beq $s0, 0, end_Delete_x
                        lw $s1, 0($s0) #Update next node
                        j checkNumber

end_Delete_x:
        move $v1, $s3 #Return head of the linked list
        move $v0, $t0 #Return the count
        lw $s0, 20 ($sp)
```

```
lw $s1, 16 ($sp)
lw $s2, 12 ($sp)
lw $s3, 8 ($sp)
lw $s4, 4 ($sp)
lw $ra, 0 ($sp)
addi $sp, $sp, 24
jr $ra
```

I can’t return the deleted node(s) back to the heap because it's impossible to free anything except for the end of the heap. In this situation, we try to return the nodes at the interior of the heap.