CS224
Section No.: 2
Spring 2018
Lab No.: 3
EFE ACER / 21602217

PRELIMINARY WORK

1. Code for the first and the second question:

```
#Prelab 3 - parts a and b
#This program recursively multiplies two numbers and finds the summation of
numbers from 1 to N.
#Author: EFE ACER

      .text
      .globl      __start

#Main
__start:
      la $a0, intro #print intro
      li $v0, 4
      syscall
      la $a0, prompt1 #print prompt to read a multiplicand
      li $v0, 4
      syscall
      li $v0, 5 #read the multiplicand
      syscall
      move $t0, $v0
      la $a0, prompt2 #print prompt to read a multiplier
      li $v0, 4
      syscall
      li $v0, 5 #read the multiplier
      syscall
      move $a0, $t0 #set $a0 to the multiplicand
      move $a1, $v0 #set $a1 to the multiplier
      jal recursiveMultiplication #call the multiplication method
      move $t0, $v0 #$t0 holds the result of the multiplication
      la $a0, result1 #print a message to display the result of the
multiplication
      li $v0, 4
      syscall
      move $a0, $t0 #print the result of the multiplication
      li $v0, 1
      syscall
      la $a0, prompt3 #print prompt to read N
      li $v0, 4
      syscall
      li $v0, 5 #read N
      syscall
      move $a0, $v0
      jal recursiveSummation #set $a0 to N
      move $t0, $v0 #$t0 holds the result of the summation
      la $a0, result2 #print a message to display the result of the
summation
      li $v0, 4
```

```
        syscall
        move $a0, $t0 #print the result of the summation
        li $v0, 1
        syscall
        li $v0, 10 #stop execution
        syscall

#This method performs a multiplication recursively
#Parameters: $a0 contains the multiplicand, $a1 contains the multiplier
#Return value: $v0 contains the result of the multiplication
recursiveMultiplication:
        subi $sp, $sp, 8
        sw $a1, 4($sp) #push (save) the multiplier to the stack
        sw $ra, 0($sp) #push (save) the return address to the stack
        bgt $a1, 1, recursiveCaseMultiplication
        #baseCase:
                move $v0, $a0 #if (multiplier == 1) {sum = multiplicand;}
                addi $sp, $sp, 8
                jr $ra #jump to the previous call
        recursiveCaseMultiplication:
                subi $a1, $a1, 1 #decrement the multiplier for recursive call
                jal recursiveMultiplication
                lw $ra, 0($sp) #pop (load) the return address from the stack
                lw $a1, 4($sp) #pop (load) the multiplier from the stack
                addi $sp, $sp, 8
                add $v0, $v0, $a0 #increment the result after recursive calls
                jr $ra #jump to the previous call

#This method finds the summation of numbers from 1 to N
#Parameters: $a0 contains N
#Return value: $v0 contains the result of the summation
recursiveSummation:
        subi $sp, $sp, 8
        sw $a0, 4($sp) #push (save) N to the stack
        sw $ra, 0($sp)    #push (save) the return address to the stack
        bgt $a0, 1, recursiveCaseSummation
        #baseCase:
                move $v0, $a0 #if (N == 1) {sum = 1;}
                addi $sp, $sp, 8
                jr $ra #jump to the previous call
        recursiveCaseSummation:
                subi $a0, $a0, 1 #decrement N for the recursive call
                jal recursiveSummation
                lw $ra, 0($sp)    #pop (load) the return address from stack
                lw $a0, 4($sp) #pop (load) N from the stack
                addi $sp, $sp, 8
                add $v0, $v0, $a0 #increment the result after recursive calls
                jr $ra #jump to the previous call

#The data segment
                .data
intro:          .asciiz     "This program recursively multiplies two numbers
and finds the summation of numbers from 1 to N.\n"
prompt1:        .asciiz     "\nPlease enter a value for the multiplicand: "
prompt2:        .asciiz     "\nPlease enter a value that is greater than 1 for
the multiplier: "
prompt3:        .asciiz     "\n\nPlease enter a value that is greater than 1
for N: "
result1:        .asciiz     "\nThe result of the multiplication is: "
```

```
result2:    .asciiz     "\nThe result of the summation is: "
```

2. Code for the third question:

```
#Prelab 3 - part c
#Deletes an element from the linked list with value x
#Parameter(s): $a0 contains the pointer to the linked list, $a1 contains x
#Return value(s): returns 0 in $v0 if deletion is successful, -1 if not,
#              returns a pointer to the head of the list in $v1
#NOTE: The program is not able to return the deleted node back to the heap,
#      since there is no heap dealocation in MARS MIPS simulator.
#      Unlike java or C++, the simulator does not provide an instruction or
#      call to avoid memory leaks.
#Author: EFE ACER
Delete_x:
      move $v1, $a0
      beq $a0, $zero, fail #deletion fails for an empty list
      lw $t1, 4($a0) #$t1 is the value in the head node
      beq $t1, $a1, deleteHead
      move $t0, $a0 #$t0 is the previous node
      searchForValue:
            lw $t1, 0($t0) #$t1 is the current node
            beq $t1, $zero, fail #failed to find the value
            lw $t2, 4($t1) #$t2 is the value in the current node
            beq $t2, $a1, deleteNext #item is found
            move $t0, $t1
            j searchForValue
      deleteHead:
            lw $v1, 0($a0) #$v1 points to the node after the previous head
            li $v0, 0 #deletion is successful
            j return
      deleteNext:
            lw $t2, 0($t1) #the next of the current node is stored in $t2
            sw $t2, 0($t0) #the next of the previous node becomes $t2
            li $v0, 0 #deletion is successful
            j return
      fail:
            li $v0, -1 #unsuccessful deletion
      return:
            jr $ra #return to caller
```