**Part 1. Preliminary Work / Preliminary Design Report (30 points)**

CS224

Section No.: 05

Spring 2018

Lab No.: 03

Asuman Aydın/21502604

**For the linked list parts only upload the utilities that you have implemented do not upload the parts provided to you.**

**1**. **(10 points) recursiveMultiplication**: Write a recursive MIPS subprogram that performs multiplication of two positive numbers by successive additions. Note that 5 X 4= 5 + 5 + 5 + 5. Error check is optional. For **recursiveMultiplication** and **recursivesummation** provide a simple user interface to test them under the same main program..

.data

x: .word 20

result: .word 0

space : .asciiz " "

newLine : " \n"

askSize: .asciiz "n:"

.text

.globl main

main:

li $v0, 4 # print askSize

la $a0, askSize

syscall

li $v0, 5

syscall

move $a0, $v0

jal recursiveSummation

sw $v0, result

#print sum

li $v0, 4

la $a0, result

syscall

li $v0, 4 # print askSize

la $a0, askSize

syscall

li $v0, 5

syscall

move $a0, $v0

li $v0, 4 # print askSize

la $a0, askSize

syscall

li $v0, 5

syscall

move $a1, $v0

jal recursiveMultiplication

sw $v0, result

#print sum

li $v0, 4

la $a0, result

syscall

recursiveMultiplication:

addi $sp, $sp, -12

sw $a1, 8($sp)

sw $a0, 4($sp)

sw $ra, 0($sp)

addi $t0, 0, 2

slt $t0, $a0, $t0 #base case

beq $t0, $0, multi

addi $v0, $0, 1

addi $sp, $sp, 8

jr $ra #return

multi:

addi $a0, $a0, 0

jal recursiveMultiplication

lw $a1, 8($sp)

lw $a0, 4($sp)

add $v0, $a0, $v0

lw $ra, 0($sp)

addi $sp, $sp, 8

jr $ra

**2**. **(10 points) recursiveSummation**: Write a recursive MIPS subprograms that finds the summation of numbers from 1 to N. Error check is optional.

recursiveSummation:

addi $sp, $sp, -8

sw $a0, 4($sp)

sw $ra, 0($sp)

addi $t0, 0, 2

slt $t0, $a0, $t0 #base case

beq $t0, $0, sum

addi $v0, $0, 1

addi $sp, $sp, 8

jr $ra #return

sum:

addi $a0, $a0, -1

jal recursiveSummation

lw $a0, 4($sp)

add $v0, $a0, $v0

lw $ra, 0($sp)

addi $sp, $sp, 8

jr $ra

**3**. **Delete\_x (10 points)**: Study the linked list program provided and the linked list explanation provided below in Part 3. Delete an element from the linked list with value x: the pointer to the linked list is passed in $a0, and the integer value of the element to be deleted is given in $a1. If there is more than one element containing the value x, the first occurrence of the value is the element to be deleted.  Return value in $v0 =0 if successful, -1 if not. In either case, the return value in $v1 contains the pointer to the head of the linked list. Are you able to return the deleted node back to the heap? If not include a comment in the program to explain why. Modify the given linked list program to include this option.

Delete\_x:

#it takes previous node and checks if it is head node or not

#then loads the next node and checks if it is tail node

#takes the address of next node and puts previous one to next

#then gets the next and previous node and stores the next one in previous one

#so it is replaced with next and deleted

#### you cannot return the deleted value to heap because we free the memory that was containing the deleted node

beqz $a0, Finish90

lw $t2, -4($a1)

beqz $t2, $a0

lw $t3, 12($a1)

beqz $t3, $a0

lw $t3, 12($a1)

sw $t2, -4($t3)

lw $t2, 12($a1)

lw $t3, -4($a1)

sw $t2, 12($t3)

la $s0, ($t2)

li $v0, -1

move $v1, $a0

jr $ra