PRELIMINARY REPORT

Lab 03

MUHAMMAD ARHAM KHAN

21701848 - CS

SECTION 06

SPRING 2019

Dated: 11 March, 2019

A- recursiveDivision

```
.text
     recursiveDivision:
          #importing arguments
          move  $t0, $a0
          move  $t1, $a1
          #checking if b is 0, to avoid error
          beq   $t1, 0, return0
          #checking if a == b so to return 1
          sub   $t2, $t0, $t1
          beq   $t2, 0, return1
          #checking if a is less than b so zero maybe returned
          blt   $t0, $t1, return0

          sub   $t0, $t0, $t1
          #counting quotient
          add   $s1, $s1, $t1
          #saving value of ra to stack
          addi  $sp, $sp, -4
          sw    $ra, 0( $sp)
          #calling the function recursively with a = a - b and b = b
          move  $a0, $t0
          move  $a1, $t1
          jal   recursiveDivision

          #retrieving returned values
          addi  $v1, $v1, 1
          lw    $ra, 0( $sp)
          addi  $sp, $sp, 4
          jr    $ra

          return0:
                add   $v1, $0, $0
                jr    $ra
           return1:
                addi  $v1, $0, 1
                #counting quotient
                add   $s1, $s1, $t1
                jr    $ra

.data

divisionPrompt:       .asciiz "Answer: "
remainderPrompt:.asciiz    "\nRemainder: "
newline:         .asciiz    "\n"
multiple:        .asciiz "multiplication: "
```

B- MultiplyDigits

```
multiplyDigits:
           #saving arguments
           move  $s0, $a0
           #checking if last digit
           blt   $s0, 10, endMultiply
           #finding the last character
           div   $s0, $s0, 10
           mfhi  $s1
           #saving current data to stack before recursion
           addi  $sp, $sp, -8
           sw    $ra, 0( $sp)
           sw    $s1, 4( $sp)
           #calling recursively with num = num/10
           move  $a0, $s0
           jal   multiplyDigits
           #loading previous data from stack
           lw    $ra, 0( $sp)
           lw    $s1, 4( $sp)
           addi  $sp, $sp, 8
           #multiplying values collected
           mul   $v1, $v1, $s1
           jr    $ra
           endMultiply:
                move  $v1, $s0
                jr    $ra
.data

divisionPrompt:       .asciiz "Answer: "
remainderPrompt: .asciiz    "\nRemainder: "
newline:         .asciiz    "\n"
multiple:        .asciiz "multiplication: "
```

C- Delete – x

```
Delete_x:
      #include arguments
      move  $s0, $a0
      move  $s1, $a1
      #save previous values to stack
      addi  $sp, $sp, -8
      sw    $s0, 0( $sp)
      sw    $ra, 4( $sp)
      #initialize previous pointer
      add   $s3, $0, $0
      loopThrough:
            #if last element reached
            beq   $s0, $0, endTraverse

            jal   checkDelete
            #increment pointer to next value
            lw    $t5, 0( $s0)
            move  $s0, $t5
            j     loopThrough

      checkDelete:
            #if current' number is to be deleted
            lw    $t2, 4($s0)
            beq   $t2, $s1, deleteThisPointer
            #increment previous node counter
            move  $s3, $s0
            jr    $ra

      deleteThisPointer:
            add   $v0, $0, $0
```

```
        #check if head deleted
        lw   $t4, 0( $sp)
        beq  $t4, $s0, headDeleted

        #the memory can be deallocated back to the heap using
        specific indexing and syscall 9 (sbrk).

        #change previous nodes pointer to next node
        lw   $t1, 0( $s0)
        sw   $t1, 0( $s3)
        jr   $ra
    headDeleted:
        #if its is head, update value in stack
        lw   $t1, 0( $s0)
        sw   $t1, 0( $sp)
        jr   $ra

    endTraverse:
        #retrieve values from stack
        lw   $s0, 0( $sp)
        lw   $ra, 4( $sp)
        addi $sp, $sp, 8
        #save values to return
        move $v1, $s0
        jr   $ra
```

Q. Are you able to return the deleted node(s) back to the heap? If not include a comment in the program to explain why.

Ans. the memory can be deallocated back to the heap using specific indexing and syscall 9 (sbrk).