Question 1: (25 points)

Consider the MIPS assembly code of the function `mysteryProc` given below. Assume that this implementation adheres to the MIPS procedure-calling conventions. Also, note that only function-local variables are stored in $\$s_x$ registers. For simplicity, the MIPS code for storing and restoring callee-saved registers to and from the stack are omitted.

```
(1)  0x0040 0000     mysteryProc: addi  $t1, $zero, 32
(2)  0x0040 0004                  sllv  $s0, $s0, $t1
(3)  0x0040 0008                   L1:  add   $t2, $a0, $zero
(4)  0x0040 000C                        lbu   $t3, 0($t2)
(5)  0x0040 0010                        bne   $t3, $zero, L2
(6)  0x0040 0014                        j     L3
(7)  0x0040 0018                   L2:  addi  $a0, $a0, 1
(8)  0x0040 001C                        addi  $s0, $s0, 1
(9)  0x0040 0020                        j     L1
(10) 0x0040 0024                   L3:  add   $v0, $zero, $s0
(11) 0x0040 0028                        jr    $ra
```

a. (5 points) How many parameters does the function `mysteryProc` have? Give a name for each parameter. You will use these names in your source code for part c of this question. Also, indicate the type of each parameter, i.e., whether it is an address or a value. Justify your answer.

b. (10 points) The MIPS implementation of `mysteryProc` given above is intentionally naïve and is not the best written code. Optimize this code to implement the same functionality but using as few and/or higher-performing MIPS instructions as possible.

c. (10 points) In class, we looked at several examples of mapping C-style high-level code to MIPS assembly code. In this question, your task is to reverse engineer the assembly code of the function `mysteryProc` to provide a C-style code that best represents the given code. The code is printed with line numbers to facilitate referencing to instructions in your answer. Note that you may find it easier to reverse engineer your optimized code rather than the given code. In a single sentence, write down what this function does.