COMP.2030 LAB 4 10/4/21

NAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In the C, the following linked list node is declared:

struct node { double val;

struct node \*next; },

Three pointers are declared by “struct node \*t1, \*t2, \*t3,” and t1 and t2 point to two successive nodes in the list. \*t3 is an pointer to an arbitrary link node. Assume that all links are NOT null pointers and point to valid addresses, and that MIPS registers $t1, $t2, and $t3 hold these C pointers t1, t2, and t3, respectively.

lw $t4, 8($t1)

sw $t4, 8($t3)

sw $t3, 8($t1)

Write the equivalent operation of the MIPS code on the right in **two** C statements.

t3->next = t1->next;

t1->next = t3;

# argument p is passed via $a0

prob: beq $a0, $zero, ret

lw $t0, ($a0)

addi $t0, $t0, 1

sw $t0, ($a0)

ret: jr $ra

1. The C function declared as

void prob (int \*p){ … }

is compiled to MIPS codes on the right.

Write the C code that performs the equivalent operation for the function ‘prob’ in **no more than two** C statements.

void prob (int \*p) {

1. Typical Boolean operators in a high level language are “and” (&), “or” (|), “exclusive or” (^), and “not” (~). A Boolean operator works on all individual bits (a bit-wise operation) to apply the same logical operation to all (pairs of) bits. Write the result of these boolean operations:

01010101 10011001 11010011

& 01101001 | 01011110 ^ 01110110 ~ 01101011

01000001 11011111 10100101 10010100

1. Suppose that x and y have byte values 0x66 and 0x39, respectively. Compare the difference between the bit-wise boolean operations (&, |, ~) versus the holistic logical operations (&&, ||, ~) by filling in the following table indicating the byte values of the C expressions:

Expression Value Expression Value

x & y \_0x20\_\_\_\_ x && y \_0x01\_\_\_

x | y \_0x7F\_\_\_\_ x || y \_0x01\_\_\_

~x|~y \_0xDF\_\_\_\_ !x||!y \_0x00\_\_\_

x & !y \_0x00\_\_\_\_ x && ~y \_0x01\_\_\_