Variant 2

6.1903: Quiz 3 (11/16/2026)

Name:	**Kerberos:**

Enter your answers in the boxes designated for each problem. Additional scratch paper can be provided but will not be considered in grading.

`app_main` calls `mystery_sum(a, b)`, shown on the next page, with two starting values that you are not told. The call to `mystery_sum` in `app_main` occurs at instruction `0x42008268`. The base case calls `mystery_bottom()`, which prints the stack at the moment it was called, starting at the current `sp` and going toward higher addresses. Study the code and stack snapshot on the next page and answer the questions below. (Feel free to separate the second page since we will only grade the first page.)

1. What two values `(a, b)` did `app_main` pass to `mystery_sum`? Give them in hexadecimal.

`a = 0x`	`b = 0x`

2. How many times had `mystery_sum` been called when the snapshot was taken, including the call that went to `mystery_base`?

3. What was the value of `sp` in `app_main` immediately before the `call mystery_sum` was executed?

`0x`

4. What is the address of the `add a0, a0, s0` instruction in `mystery_sum`?

`0x`

5. When execution reaches `mystery_base`, what values (in hexadecimal) are in registers `a0` and `s0`?

`a0 = 0x`	`s0 = 0x`

6. What value does `mystery_sum` eventually return to `app_main`? **Give your answer in decimal.** (Show work in box for partial credit.)

Variant 2

Feel free to separate this page from the first page! We will not be scanning it for grading.

```
mystery_sum:
    li    t0, 100
    bgt   a0, t0, mystery_base  # a > limit: done
    addi  sp, sp, -12
    sw    ra, 8(sp)
    sw    s0, 4(sp)
    sw    a0, 0(sp)
    mv    s0, a0
    add   t1, a0, a1
    mv    a0, a1
    mv    a1, t1
    call  mystery_sum
    add   a0, a0, s0 # <--question 4
    lw    s0, 4(sp)
    lw    ra, 8(sp)
    addi  sp, sp, 12
    ret
mystery_base:
    addi  sp, sp, -4
    sw    ra, 0(sp)
    call  mystery_bottom # stack snapshot here
    lw    ra, 0(sp)
    addi  sp, sp, 4
    li    a0, 0
    ret
```

```
0x3fc9b87c: 0x42007d48
0x3fc9b880: 0x0000004c
0x3fc9b884: 0x0000002f
0x3fc9b888: 0x42007d48
0x3fc9b88c: 0x0000002f
0x3fc9b890: 0x0000001d
0x3fc9b894: 0x42007d48
0x3fc9b898: 0x0000001d
0x3fc9b89c: 0x00000012
0x3fc9b8a0: 0x42007d48
0x3fc9b8a4: 0x00000012
0x3fc9b8a8: 0x0000000b
0x3fc9b8ac: 0x42007d48
0x3fc9b8b0: 0x0000000b
0x3fc9b8b4: 0x00000007
0x3fc9b8b8: 0x42007d48
0x3fc9b8bc: 0x00000007
0x3fc9b8c0: 0x00000004
0x3fc9b8c4: 0x42007d48
0x3fc9b8c8: 0x00000004
0x3fc9b8cc: 0x00000003
0x3fc9b8d0: 0x42007d48
0x3fc9b8d4: 0x00000003
0x3fc9b8d8: 0x0000000d
0x3fc9b8dc: 0x4200826c
0x3fc9b8e0: 0x00000001
0x3fc9b8e4: 0x3fc8f210
0x3fc9b8e8: 0x3fc9b930
0x3fc9b8ec: 0x42008b98
```