Variant 3

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 `0x4200af0c`. 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 3

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
```

```
0x3fc9d23c: 0x4200a3e4
0x3fc9d240: 0x00000049
0x3fc9d244: 0x0000002d
0x3fc9d248: 0x4200a3e4
0x3fc9d24c: 0x0000002d
0x3fc9d250: 0x0000001c
0x3fc9d254: 0x4200a3e4
0x3fc9d258: 0x0000001c
0x3fc9d25c: 0x00000011
0x3fc9d260: 0x4200a3e4
0x3fc9d264: 0x00000011
0x3fc9d268: 0x0000000b
0x3fc9d26c: 0x4200a3e4
0x3fc9d270: 0x0000000b
0x3fc9d274: 0x00000006
0x3fc9d278: 0x4200a3e4
0x3fc9d27c: 0x00000006
0x3fc9d280: 0x00000005
0x3fc9d284: 0x4200a3e4
0x3fc9d288: 0x00000005
0x3fc9d28c: 0x00000001
0x3fc9d290: 0x4200a3e4
0x3fc9d294: 0x00000001
0x3fc9d298: 0x00000009
0x3fc9d29c: 0x4200af10
0x3fc9d2a0: 0x3fc8a604
0x3fc9d2a4: 0x00000000
0x3fc9d2a8: 0x3fc9d310
0x3fc9d2ac: 0x4200b6f0
```