Quiz 1 (Sep 9, 2025)

Name: ANSWERS NetID (e.g., ap191):

Please write your answers clearly and keep them brief.

Problem	Score
Question 1	/ 5
Question 2	/ 5
Total	/ 10

1. The listing in handout 1 (which you received last lecture) indicated that there was a bug in function `g`. We have reproduced the necessary bit of code below, explain (briefly) the bug.

```
1  #include <stdio.h>
2  #include <stdint.h>
3
4  uint64_t *q;
5
6  /* ... */
7
8  uint64_t g(uint64_t a) {
9      uint64_t x = 2 * a;
10     q = &x; // <-- THIS IS AN ERROR (AKA BUG)
11     return x;
12 }
```

The line sets the global pointer `q` to point to the location of a local variable `x` which is a bug.

To understand why in our context, observe that `x` is a local variable and is thus located within the stack frame for `g`'s current invocation. The line sets `q` to the address of this location in the stack. However, when `g` returns the space occupied by its stack frame is marked as unused (remember the epilog resets `%rsp` and `%rbp` to the values from before `g`'s prolog was run). This means that other functions called after `g` might reuse the memory location for other values, including different local variable or even return addresses. Thus `*q`'s value (i.e., the value pointed to by `q`) can change unexpectedly after `g` returns.

2. Write code within the function `g` (in the place marked your code goes here) so that the register `%rax` contains the beginning of its caller's stack frame. For example, when `g` is called by `f`, your code snipet should result in the beginning of `f`'s stackframe being stored in `%rax`.

   You should assume that `f` and `g` have the standard prolog and epilog we discussed last class. Drawing `g`'s stack frame is likely to help you arrive at the answer.

```
1  void f() {
2     uint64_t frame_begin = fun();
3  }
4
5  uint64_t g() {
6     /* You can use assembly or C.
7        Assembly reminder:
8          movq src, dst
9          Registers: %rbx <- the value of %rbx
10         Pointer deref (%rbx) <- the value at the address in %rbx
11       YOUR LOGIC GOES HERE
12    */
13 }
```

   `movq (%rbp), %rax`

   [Note, we give full credit if the answer is accompanied by a stack frame that has %rbp in a slightly different location (e.g., one stack slot above).]