$\boxed{3.70}$ e1 → 16 bytes

e2 → 16 bytes

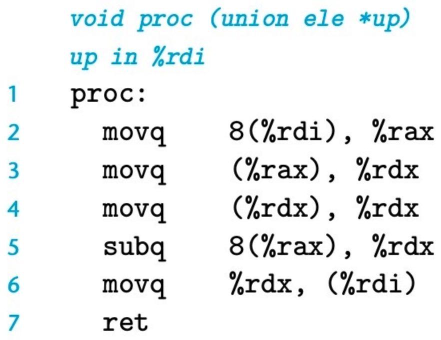
A) e1.p= 0, e1.x= 8, e2.y=0, e2.next= 8

B) 16 bytes

C) void proc(union ele *up) {

up → e2.x = *(up → e2.next → e1.p) - up→e1.y

}



```
void proc (union ele *up)
up in %rdi
1   proc:
2     movq    8(%rdi), %rax
3     movq    (%rax), %rdx
4     movq    (%rdx), %rdx
5     subq    8(%rax), %rdx
6     movq    %rdx, (%rdi)
7     ret
```

* memory address in up stored in rdi

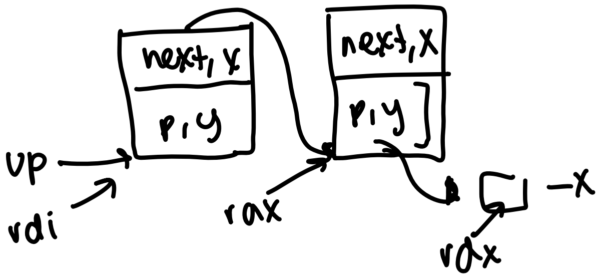
2) move up one memory address, dereference, store in rax } store next node in rax

3) deref rax, store in rdx } store p in rdx

4) deref rdx, store in rdx. } store val of p in rdx

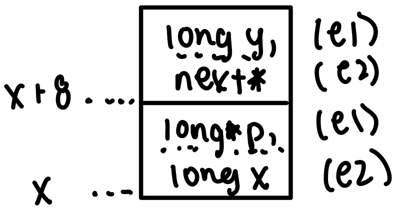
5) subtract dereference of 8+ rax in rdx

6) move rdx into memory address in rdi.



up → e1.p= e2.next →

UNION ELE



3.89

A) True. Doubles can cast integers w/o loss of accuracy, so there is no difference between casting from int to double to float and from int directly to float.

B) False. counter example:

$x = Tmin = -2^{31}$

$y = -1$

$x - y = Tmin - 1 = Tmax = 2^{31} - 1$

$dx - dy = -2^{31} - 1$

C) True, because we can save any 53-bit integer without loss of accuracy.

d) False. If $dx \approx dy$ but $dz \gg dy$, then we might lose precision doing $(dy + dz)$ that we might not have had we done $(dx + dy)$ first.

E) False if x or z is 0.