3.68

由 str1 和 str2 的定义，这两个结构的空间分配类似下面的表示

Str1

4 * A * B	4 或 0	8
x[A][B]		y

Str2

B	0~3	4	2 * A	0~6 的偶数	8
Array[B]		t	s[A]		u

则由对齐的规则有: 4 < B <= 8 (由第二行 8(%rsi)得出)

12 < 2 * A <= 20 (由第三行 32(%rsi)得出)

176 < 4 * A * B <= 184 (由第四行 184(%rdi)得出)

其中 A，B 为整数

解得 A = 9, B = 5

3.69

```
<test>:
mov     0x120(%rsi), %ecx          #last 在 bp+0x120 的位置
add     (%rsi), %rcx               #first 就在 bp 那里
lea     (%rdi, %rdi, 4), %rax      #5  * i
lea     (%rsi, %rax, 8), %rax      #40 * i + rsi,得出 a_struct 的大小为 40
mov     0x8(%rax), %rdx            #加上开始 first 和对齐的偏移量,0x8(%rax)
                                   是 ap->idx 的值,且 idx 是 long 型的整数
movslq  %ecx, %rcx                 #x 数组的元素应为 long
mov     %rcx, 0x10(%rax, %rdx, 8)  #再加上 idx 的长度,ap->x 在 0x10(%rax)
retq
```

A CNT = (0x120 - 0x8) / 40 = 7 (0x8 是因为 a_struct 的对齐要求)

B
```
typedef struct{
    long idx;
    long x[4];
}a_struct;
```

3.70

A

e1.p	0
e1.y	8
e2.x	0
e2.next	8

B 16

C

```
proc:
   movq 8(%rdi), %rax          # e1.y / e2.next   (this)
   movq (%rax), %rdx           # e1.p / e2.x     (next)
   movq (%rdx), %rdx           # *p               (next)
   subq 8(%rax), %rdx          # *p - y           (next)
   movq %rdx, (%rdi)           # x                (this)
   ret
```

右边标注了每步中,右边的寄存器可能代表的值
则可以得出 proc 缺失的表达式

```
void proc(union ele * up){
    up->e2.x = *(up->e2.next->e1.p)
               - up->e2.next->e1.y;
}
```