E. 完成 eval 的栈帧图，给出在从 process 返回后 eval 是如何访问结构 r 的元素的。

F. 就如何传递作为函数参数的结构以及如何返回作为函数结果的结构值，你可以看出什么通用的原则？

**3.68 在下面的代码中，*A* 和 *B* 是用# define 定义的常数：

```
typedef struct {
    int x[A][B]; /* Unknown constants A and B */
    long y;
} str1;

typedef struct {
    char array[B];
    int t;
    short s[A];
    long u;
} str2;

void setVal(str1 *p, str2 *q) {
    long v1 = q->t;
    long v2 = q->u;
    p->y = v1+v2;
}
```

GCC 为 setVal 产生下面的代码：

```
void setVal(str1 *p, str2 *q)
p in %rdi, q in %rsi
setVal:
  movslq  8(%rsi), %rax
  addq    32(%rsi), %rax

  movq    %rax, 184(%rdi)
  ret
```

A 和 *B* 的值是多少？（答案是唯一的。）

**3.69 你负责维护一个大型的 C 程序，遇到下面的代码：

```
typedef struct {
    int first;
    a_struct a[CNT];
    int last;
} b_struct;

void test(long i, b_struct *bp)
{
    int n = bp->first + bp->last;
    a_struct *ap = &bp->a[i];
    ap->x[ap->idx] = n;
}
```

编译时常数 CNT 和结构 a_struct 的声明是在一个你没有访问权限的文件中。幸好，你有代码的‘.o’版本，可以用 OBJDUMP 程序来反汇编这些文件，得到下面的反汇编代码：

```
void test(long i, b_struct *bp)
i in %rdi, bp in %rsi
0000000000000000 <test>:
   0:   8b 8e 20 01 00 00       mov    0x120(%rsi),%ecx
   6:   03 0e                   add    (%rsi),%ecx
   8:   48 8d 04 bf             lea    (%rdi,%rdi,4),%rax
   c:   48 8d 04 c6             lea    (%rsi,%rax,8),%rax
  10:   48 8b 50 08             mov    0x8(%rax),%rdx
  14:   48 63 c9                movslq %ecx,%rcx
```