C. `struct P3 { short w[3]; char c[3] };`

w	c	总共	对齐
0	6	10	2

D. `struct P4 { short w[5]; char *c[3] };`

w	c	总共	对齐
0	16	40	8

E. `struct P5 { struct P3 a[2]; struct P2 t };`

a	t	总共	对齐
0	24	40	8

3.45 这是一个理解结构的布局和对齐的练习。

A. 这里是对象大小和字节偏移量：

字段	a	b	c	d	e	f	g	h
大小	8	2	8	1	4	1	8	4
偏移量	0	8	16	24	28	32	40	48

B. 这个结构一共是 56 个字节长。结构的结尾必须填充 4 个字节来满足 8 字节对齐的要求。

C. 当所有的数据元素的长度都是 2 的幂时，一种行之有效的策略是按照大小的降序排列结构的元素。导致声明如下：

```
struct {
    char     *a;
    double    c;
    long      g;
    float     e;
    int       h;
    short     b;
    char      d;
    char      f;
} rec;
```

得到的偏移量如下：

字段	a	c	g	e	h	b	d	f
大小	8	8	8	4	4	2	1	1
偏移量	0	8	16	24	28	32	34	35

这个结构要填充 4 个字节以满足 8 字节对齐的要求，所以总共是 40 个字节。

3.46 这个问题覆盖的话题比较广泛，例如栈帧、字符串表示、ASCII 码和字节顺序。它说明了越界的内存引用的危险性，以及缓冲区溢出背后的基本思想。

A. 执行了第 3 行后的栈：