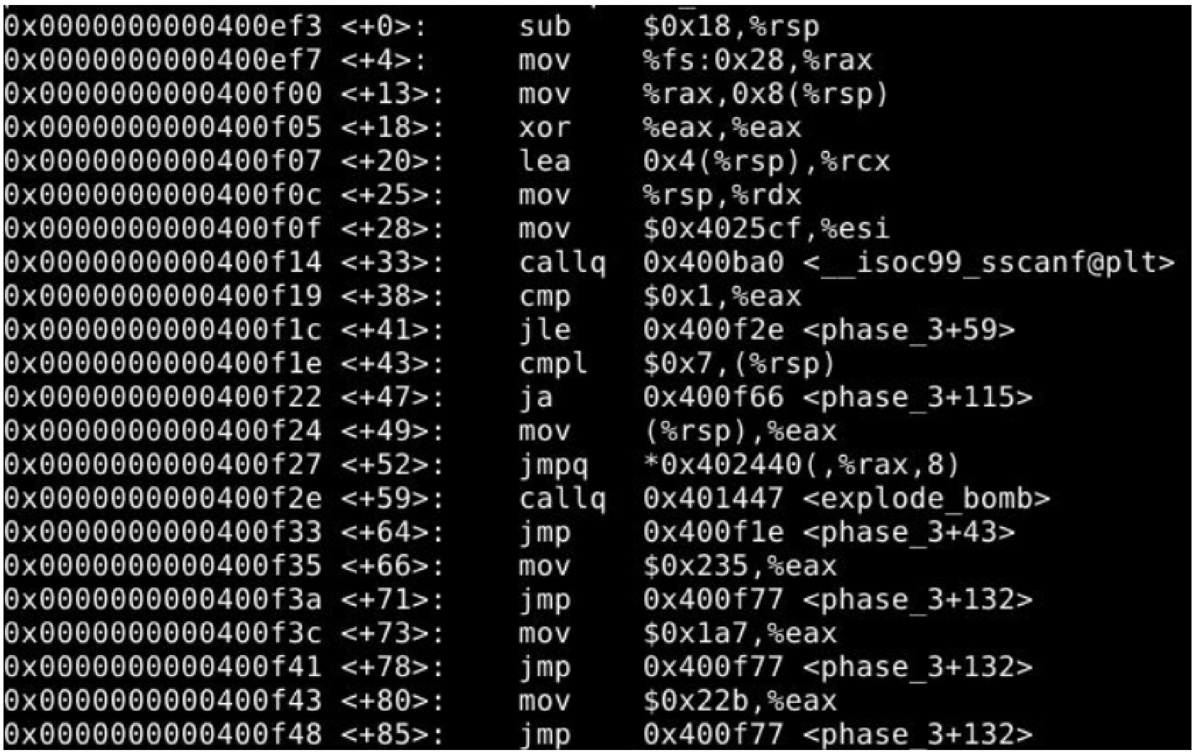
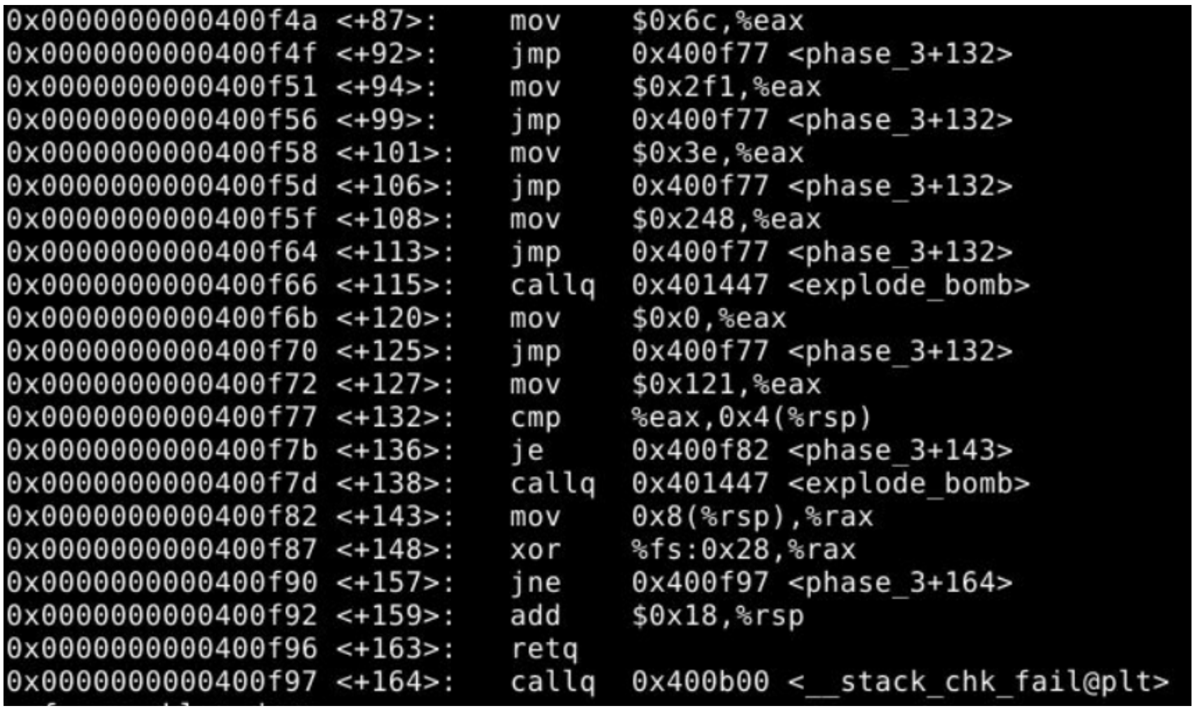
反汇编结果：

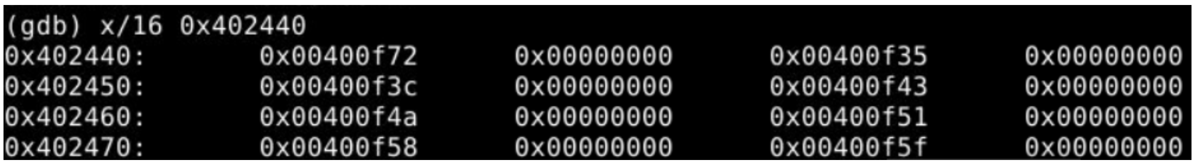


```
0x0000000000400ef3 <+0>:     sub    $0x18,%rsp
0x0000000000400ef7 <+4>:     mov    %fs:0x28,%rax
0x0000000000400f00 <+13>:    mov    %rax,0x8(%rsp)
0x0000000000400f05 <+18>:    xor    %eax,%eax
0x0000000000400f07 <+20>:    lea    0x4(%rsp),%rcx
0x0000000000400f0c <+25>:    mov    %rsp,%rdx
0x0000000000400f0f <+28>:    mov    $0x4025cf,%esi
0x0000000000400f14 <+33>:    callq  0x400ba0 <__isoc99_sscanf@plt>
0x0000000000400f19 <+38>:    cmp    $0x1,%eax
0x0000000000400f1c <+41>:    jle    0x400f2e <phase_3+59>
0x0000000000400f1e <+43>:    cmpl   $0x7,(%rsp)
0x0000000000400f22 <+47>:    ja     0x400f66 <phase_3+115>
0x0000000000400f24 <+49>:    mov    (%rsp),%eax
0x0000000000400f27 <+52>:    jmpq   *0x402440(,%rax,8)
0x0000000000400f2e <+59>:    callq  0x401447 <explode_bomb>
0x0000000000400f33 <+64>:    jmp    0x400f1e <phase_3+43>
0x0000000000400f35 <+66>:    mov    $0x235,%eax
0x0000000000400f3a <+71>:    jmp    0x400f77 <phase_3+132>
0x0000000000400f3c <+73>:    mov    $0x1a7,%eax
0x0000000000400f41 <+78>:    jmp    0x400f77 <phase_3+132>
0x0000000000400f43 <+80>:    mov    $0x22b,%eax
0x0000000000400f48 <+85>:    jmp    0x400f77 <phase_3+132>
```



```
0x0000000000400f4a <+87>:    mov    $0x6c,%eax
0x0000000000400f4f <+92>:    jmp    0x400f77 <phase_3+132>
0x0000000000400f51 <+94>:    mov    $0x2f1,%eax
0x0000000000400f56 <+99>:    jmp    0x400f77 <phase_3+132>
0x0000000000400f58 <+101>:   mov    $0x3e,%eax
0x0000000000400f5d <+106>:   jmp    0x400f77 <phase_3+132>
0x0000000000400f5f <+108>:   mov    $0x248,%eax
0x0000000000400f64 <+113>:   jmp    0x400f77 <phase_3+132>
0x0000000000400f66 <+115>:   callq  0x401447 <explode_bomb>
0x0000000000400f6b <+120>:   mov    $0x0,%eax
0x0000000000400f70 <+125>:   jmp    0x400f77 <phase_3+132>
0x0000000000400f72 <+127>:   mov    $0x121,%eax
0x0000000000400f77 <+132>:   cmp    %eax,0x4(%rsp)
0x0000000000400f7b <+136>:   je     0x400f82 <phase_3+143>
0x0000000000400f7d <+138>:   callq  0x401447 <explode_bomb>
0x0000000000400f82 <+143>:   mov    0x8(%rsp),%rax
0x0000000000400f87 <+148>:   xor    %fs:0x28,%rax
0x0000000000400f90 <+157>:   jne    0x400f97 <phase_3+164>
0x0000000000400f92 <+159>:   add    $0x18,%rsp
0x0000000000400f96 <+163>:   retq
0x0000000000400f97 <+164>:   callq  0x400b00 <__stack_chk_fail@plt>
```

x /16 0x402440的结果：



```
(gdb) x/16 0x402440
0x402440:       0x00400f72      0x00000000      0x00400f35      0x00000000
0x402450:       0x00400f3c      0x00000000      0x00400f43      0x00000000
0x402460:       0x00400f4a      0x00000000      0x00400f51      0x00000000
0x402470:       0x00400f58      0x00000000      0x00400f5f      0x00000000
```

```
rsp -= 0x18
eax = 0
rcx = rsp + 0x4
```

```
rdx = rsp
esi = 0x4025cf  （%d %d）
call <sscanf>
if(eax - 0x1 <= 0)  //eax存的是sccanf返回的参数个数，说明要求参数个数>1
    call <explode_bomb>
if(*rsp - 0x7 > 0)  // 第一个参数值<=7(网上说0x8(%rsp)才是第一个参数值，但是从实际汇编代码来看这里应该(%rsp)就是第一个参数值)
    call <explode_bomb>
eax = *rsp
跳转到0x402440 + rax * 8的位置(例如rax=1，跳转到上图中的0x400f35的位置)
if(rax == 0){
    eax = 0x121
}
if(rax == 1){
    eax = 0x235
}
if(rax == 2){
    eax = 0x1a7
}
if(rax == 3){
    eax = 0x22b
}
if(rax == 4){
    eax = 0x6c
}
if(rax == 5){
    eax = 0x2f1
}
if(rax == 6){
    eax = 0x3e
}
if(rax == 7){
    eax = 0x248
}
if(*(rsp + 0x4) - eax == 0)
    retq
```

所以这道题答案有多个，分别为：

0 289

1 565

2 423

3 555

4 108

5 753

6 62

7 584