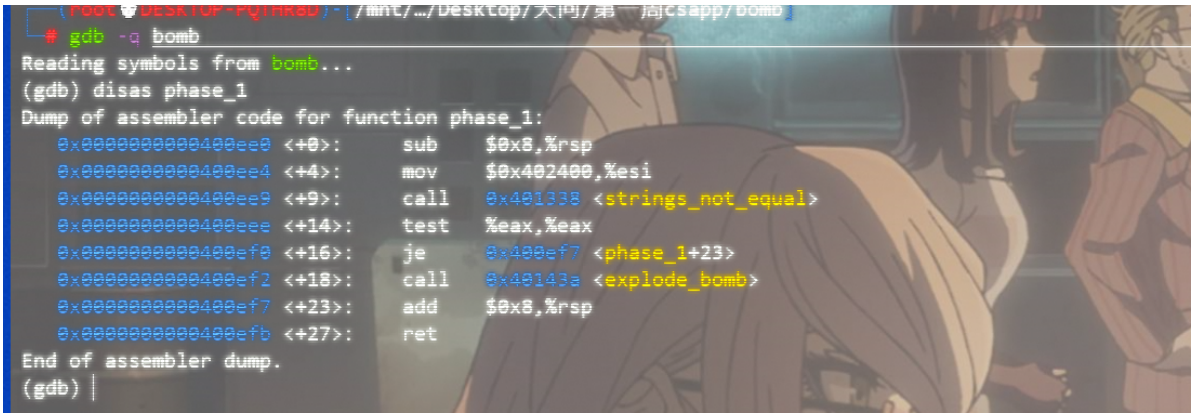
BombLab

phase_1

用gdb调



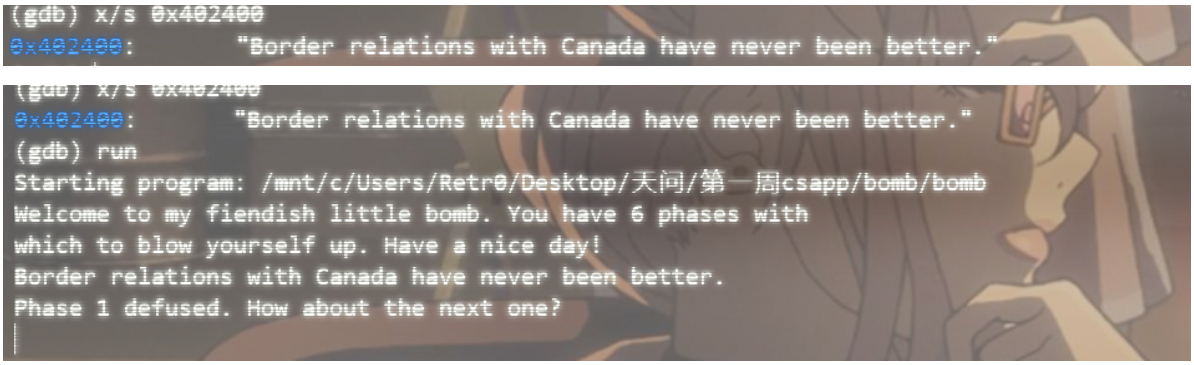

```
   0x0000000000400ee0 <+0>:     sub    $0x8,%rsp  //  rsp栈指针-8
   0x0000000000400ee4 <+4>:     mov    $0x402400,%esi //  将0x402400（可以猜的出来是后面比较字符串存放字符串的地址） 赋给%esi
   0x0000000000400ee9 <+9>:     call   0x401338 <strings_not_equal> // call 了 偏移量 0x401338的  函数
   0x0000000000400eee <+14>:    test   %eax,%eax    // 就是and 但是不会把结果保留在ax
   0x0000000000400ef0 <+16>:    je     0x400ef7 <phase_1+23> // 根据上部指令 如果eax 为0  就可以跳到 add指令。
   0x0000000000400ef2 <+18>:    call   0x40143a <explode_bomb>  // 直接bomb
   0x0000000000400ef7 <+23>:    add    $0x8,%rsp  // 把rsp栈指针加回去。
   0x0000000000400efb <+27>:    ret //return
```

那么这里能避免bomb的关键就在call strings_not_equal里面。

所以跟一下函数

利用

```
x/s 0x402400 查一下字符串。
```

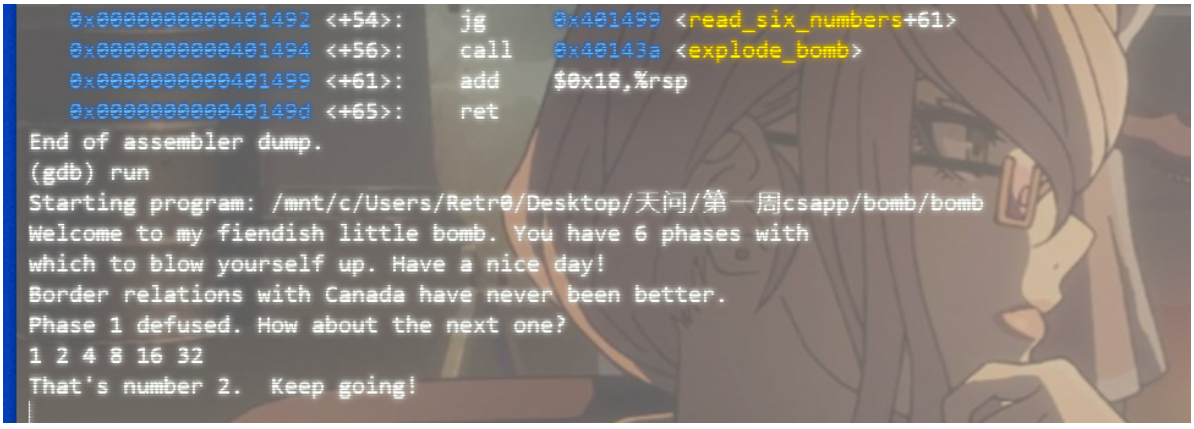


第一个phase就绕过了

```
   0x0000000000400efc <+0>:     push   %rbp    // rbp地址放入栈顶
   0x0000000000400efd <+1>:     push   %rbx  // rbx地址放栈顶
   0x0000000000400efe <+2>:     sub    $0x28,%rsp // rsp地址-0x28
   0x0000000000400f02 <+6>:     mov    %rsp,%rsi   //rsi=rsp
   0x0000000000400f05 <+9>:     call   0x40145c <read_six_numbers> // call地址。
   0x0000000000400f0a <+14>:    cmpl   $0x1,(%rsp) //  第一个数字得是1 否则就会到炸弹了。
   0x0000000000400f0e <+18>:    je     0x400f30 <phase_2+52> // jump到+52
   0x0000000000400f10 <+20>:    call   0x40143a <explode_bomb>
   0x0000000000400f15 <+25>:    jmp    0x400f30 <phase_2+52>
   0x0000000000400f17 <+27>:    mov    -0x4(%rbx),%eax  // （第一轮）将rbx -0x4 也就是 rsp 也就是 0x1 赋值给 eax
   0x0000000000400f1a <+30>:    add    %eax,%eax //  自加一波 eax+=eax
   0x0000000000400f1c <+32>:    cmp    %eax,(%rbx) // 将eax 中的值与*(rbx) 也就是*（rsp+0x4) 也就是第二个数。
   0x0000000000400f1e <+34>:    je     0x400f25 <phase_2+41>  // 相等就跳转，否则就会GG
   0x0000000000400f20 <+36>:    call   0x40143a <explode_bomb>
   0x0000000000400f25 <+41>:    add    $0x4,%rbx // %rbx+=4
   0x0000000000400f29 <+45>:    cmp    %rbp,%rbx // %rbp和rbx也就是不相等这里就变成循环节了，相等就可以了。
   0x0000000000400f2c <+48>:    jne    0x400f17 <phase_2+27>
   0x0000000000400f2e <+50>:    jmp    0x400f3c <phase_2+64> // 跳出了 也就是躲过bomb了
   0x0000000000400f30 <+52>:    lea    0x4(%rsp),%rbx //把 %rsp+0x4赋值给rbx
   0x0000000000400f35 <+57>:    lea    0x18(%rsp),%rbp // 把 %rsp+0x18 赋值给rbp
   0x0000000000400f3a <+62>:    jmp    0x400f17 <phase_2+27> // 跳转到27
   0x0000000000400f3c <+64>:    add    $0x28,%rsp
   0x0000000000400f40 <+68>:    pop    %rbx
   0x0000000000400f41 <+69>:    pop    %rbp
   0x0000000000400f42 <+70>:    ret
```

1 2 4 8 16 32

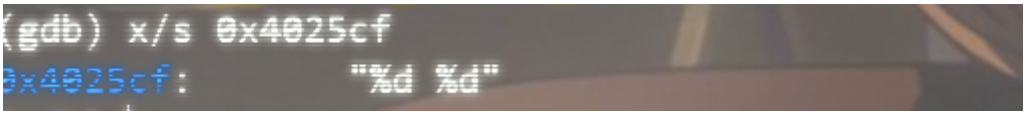
序列应该如上。



第三题

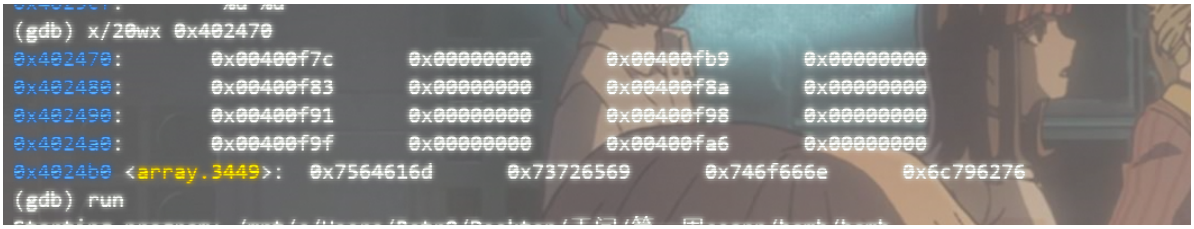
```
0x0000000000400f43 <+0>:     sub    $0x18,%rsp
   0x0000000000400f47 <+4>:     lea    0xc(%rsp),%rcx // %rsp+0xc赋值给 rcx
   0x0000000000400f4c <+9>:     lea    0x8(%rsp),%rdx // rsp+0x8 赋值给 rdx
```

```
   0x0000000000400f51 <+14>:    mov    $0x4025cf,%esi   // 将0x4025cf地址(应该是用来
comp的东西) 赋值给%esi
   0x0000000000400f56 <+19>:    mov    $0x0,%eax // 把 0x0 地址 赋值给%eax
   0x0000000000400f5b <+24>:    call   0x400bf0 <__isoc99_sscanf@plt> //call
scanf 也就是输入
   0x0000000000400f60 <+29>:    cmp    $0x1,%eax  //  eax>1 才能跳否则就要bomb了
   0x0000000000400f63 <+32>:    jg     0x400f6a <phase_3+39> //跳到39
   0x0000000000400f65 <+34>:    call   0x40143a <explode_bomb>
   0x0000000000400f6a <+39>:    cmpl   $0x7,0x8(%rsp)  //  *(rsp+0x8)  < 7 接着跳
转。
   0x0000000000400f6f <+44>:    ja     0x400fad <phase_3+106> // jump 到106
   0x0000000000400f71 <+46>:    mov    0x8(%rsp),%eax // eax = *(rsp+0x8)
   0x0000000000400f75 <+50>:    jmp    *0x402470(,%rax,8) //
   0x0000000000400f7c <+57>:    mov    $0xcf,%eax
   0x0000000000400f81 <+62>:    jmp    0x400fbe <phase_3+123>
   0x0000000000400f83 <+64>:    mov    $0x2c3,%eax
   0x0000000000400f88 <+69>:    jmp    0x400fbe <phase_3+123>
   0x0000000000400f8a <+71>:    mov    $0x100,%eax
   0x0000000000400f8f <+76>:    jmp    0x400fbe <phase_3+123>
   0x0000000000400f91 <+78>:    mov    $0x185,%eax
   0x0000000000400f96 <+83>:    jmp    0x400fbe <phase_3+123>
   0x0000000000400f98 <+85>:    mov    $0xce,%eax
   0x0000000000400f9d <+90>:    jmp    0x400fbe <phase_3+123>
   0x0000000000400f9f <+92>:    mov    $0x2aa,%eax
   0x0000000000400fa4 <+97>:    jmp    0x400fbe <phase_3+123>
   0x0000000000400fa6 <+99>:    mov    $0x147,%eax
   0x0000000000400fab <+104>:   jmp    0x400fbe <phase_3+123>
   0x0000000000400fad <+106>:   call   0x40143a <explode_bomb>  //直接bomb。
   0x0000000000400fb2 <+111>:   mov    $0x0,%eax
   0x0000000000400fb7 <+116>:   jmp    0x400fbe <phase_3+123>
   0x0000000000400fb9 <+118>:   mov    $0x137,%eax
   0x0000000000400fbe <+123>:   cmp    0xc(%rsp),%eax // eax= *(rsp+0xc)
   0x0000000000400fc2 <+127>:   je     0x400fc9 <phase_3+134>
   0x0000000000400fc4 <+129>:   call   0x40143a <explode_bomb>
   0x0000000000400fc9 <+134>:   add    $0x18,%rsp
   0x0000000000400fcd <+138>:   ret
```



查了下这个位置 是输入了2个%d

jump *实现的是一个跳转表



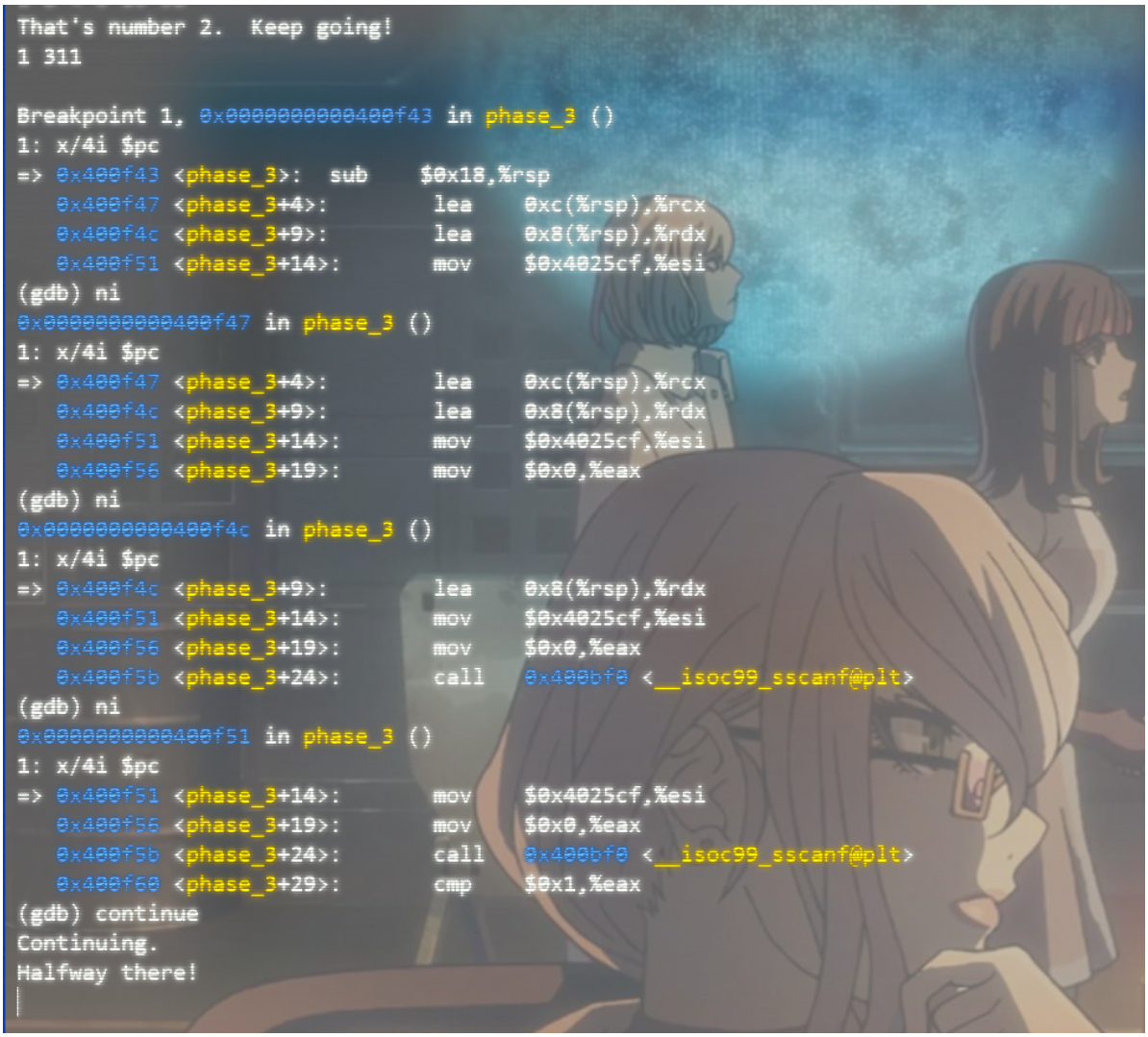
这里输入1 可以跳转到 400fb9

```
  0x0000000000400fb9 <+118>:    mov    $0x137,%eax
```

这里赋值0x137 然后cmp了一波

进行了一次cmp 也就是两次相等即可。

1 311



第四题

```
 0x000000000040100c <+0>:     sub    $0x18,%rsp
   0x0000000000401010 <+4>:     lea    0xc(%rsp),%rcx //第二个输入
   0x0000000000401015 <+9>:     lea    0x8(%rsp),%rdx //第一个输入
   0x000000000040101a <+14>:    mov    $0x4025cf,%esi  // %d %d
   0x000000000040101f <+19>:    mov    $0x0,%eax
   0x0000000000401024 <+24>:    call   0x400bf0 <__isoc99_sscanf@plt> //调用scanf
输入2个
   0x0000000000401029 <+29>:    cmp    $0x2,%eax // eax不等于2跳转
   0x000000000040102c <+32>:    jne    0x401035 <phase_4+41>
   0x000000000040102e <+34>:    cmpl   $0xe,0x8(%rsp) // *（rsp+0x8)<14
   0x0000000000401033 <+39>:    jbe    0x40103a <phase_4+46>
   0x0000000000401035 <+41>:    call   0x40143a <explode_bomb> // bomb
   0x000000000040103a <+46>:    mov    $0xe,%edx //edx= 14
   0x000000000040103f <+51>:    mov    $0x0,%esi//esi=0
   0x0000000000401044 <+56>:    mov    0x8(%rsp),%edi //edi=*(rsp+0x8)
   0x0000000000401048 <+60>:    call   0x400fce <func4>  // call func4
   0x000000000040104d <+65>:    test   %eax,%eax // 不等于0跳转
   0x000000000040104f <+67>:    jne    0x401058 <phase_4+76> //跳到bomb
   0x0000000000401051 <+69>:    cmpl   $0x0,0xc(%rsp) // 0和 *(rsp+0xc)
   0x0000000000401056 <+74>:    je     0x40105d <phase_4+81> // 相等跳 过bomb 成功
绕过
```

```
   0x0000000000401058 <+76>:    call   0x40143a <explode_bomb>
   0x000000000040105d <+81>:    add    $0x18,%rsp
   0x0000000000401061 <+85>:    ret
```

反汇编一下func4

```
  0x0000000000400fce <+0>:     sub    $0x8,%rsp
   0x0000000000400fd2 <+4>:     mov    %edx,%eax //eax=edx
   0x0000000000400fd4 <+6>:     sub    %esi,%eax //esi-=eax
   0x0000000000400fd6 <+8>:     mov    %eax,%ecx //ecx=eax
   0x0000000000400fd8 <+10>:    shr    $0x1f,%ecx
   0x0000000000400fdb <+13>:    add    %ecx,%eax
   0x0000000000400fdd <+15>:    sar    %eax
   0x0000000000400fdf <+17>:    lea    (%rax,%rsi,1),%ecx
   0x0000000000400fe2 <+20>:    cmp    %edi,%ecx // edi<=ecx就能跳出
   0x0000000000400fe4 <+22>:    jle    0x400ff2 <func4+36> //
   0x0000000000400fe6 <+24>:    lea    -0x1(%rcx),%edx
   0x0000000000400fe9 <+27>:    call   0x400fce <func4>
   0x0000000000400fee <+32>:    add    %eax,%eax
   0x0000000000400ff0 <+34>:    jmp    0x401007 <func4+57>
   0x0000000000400ff2 <+36>:    mov    $0x0,%eax //eax=0
   0x0000000000400ff7 <+41>:    cmp    %edi,%ecx //edi>=ecx?
   0x0000000000400ff9 <+43>:    jge    0x401007 <func4+57>
   0x0000000000400ffb <+45>:    lea    0x1(%rcx),%esi
   0x0000000000400ffe <+48>:    call   0x400fce <func4>
   0x0000000000401003 <+53>:    lea    0x1(%rax,%rax,1),%eax
   0x0000000000401007 <+57>:    add    $0x8,%rsp
   0x000000000040100b <+61>:    ret
```

所以在移位之后 我们发现只要输入0 就可以满足func4跳出。

```
   0x000000000040104d <+63>:    test   %eax,%eax // 不等于0跳转
   0x000000000040104f <+67>:    jne    0x401058 <phase_4+76> //跳到bomb
   0x0000000000401051 <+69>:    cmpl   $0x0,0xc(%rsp) // 0和 *(rsp+0xc)
   0x0000000000401056 <+74>:    je     0x40105d <phase_4+81> // 相等跳 过bomb 成功绕过
```

这里也需要等于0

所以输入 0 0 通过第四关

第五题

```
   0x0000000000401062 <+0>:     push   %rbx
   0x0000000000401063 <+1>:     sub    $0x20,%rsp //rsp-=20
   0x0000000000401067 <+5>:     mov    %rdi,%rbx  //rbx=rdi
   0x000000000040106a <+8>:     mov    %fs:0x28,%rax // rax =40
   0x0000000000401073 <+17>:    mov    %rax,0x18(%rsp)  // *(rsp+0x18) = rax
   0x0000000000401078 <+22>:    xor    %eax,%eax // eax=0
   0x000000000040107a <+24>:    call   0x40131b <string_length>  // 查输入长度 必须
是6
   0x000000000040107f <+29>:    cmp    $0x6,%eax  // eax=6 防止Bomb
   0x0000000000401082 <+32>:    je     0x4010d2 <phase_5+112>  //跳 112
   0x0000000000401084 <+34>:    call   0x40143a <explode_bomb>
   0x0000000000401089 <+39>:    jmp    0x4010d2 <phase_5+112>

   0x000000000040108b <+41>:    movzbl (%rbx,%rax,1),%ecx   //ecx=*(rbx+rax)
```

```
   0x000000000040108f <+45>:    mov    %cl,(%rsp) // *(rsp)=cl (ecx 低八位)
   0x0000000000401092 <+48>:    mov    (%rsp),%rdx //rdx= *(rsp)
   0x0000000000401096 <+52>:    and    $0xf,%edx // edx&=15
   0x0000000000401099 <+55>:    movzbl 0x4024b0(%rdx),%edx  // edx=*
(rdx+0x4024b0)
   0x00000000004010a0 <+62>:    mov    %dl,0x10(%rsp,%rax,1) // *
(rsp+rax+0x10)=%dl
   0x00000000004010a4 <+66>:    add    $0x1,%rax // rax+=1
   0x00000000004010a8 <+70>:    cmp    $0x6,%rax // rax=6
   0x00000000004010ac <+74>:    jne    0x40108b <phase_5+41> //循环节。 6次。
   0x00000000004010ae <+76>:    movb   $0x0,0x16(%rsp)  // *(rsp+0x16)=0
   0x00000000004010b3 <+81>:    mov    $0x40245e,%esi // esi=0x40245e (flyers)
   0x00000000004010b8 <+86>:    lea    0x10(%rsp),%rdi // rdi=*(rsp+0x10)
   0x00000000004010bd <+91>:    call   0x401338 <strings_not_equal> // 查是否相同
   0x00000000004010c2 <+96>:    test   %eax,%eax   //后面不用看了 相同就通了。
   0x00000000004010c4 <+98>:    je     0x4010d9 <phase_5+119>
   0x00000000004010c6 <+100>:   call   0x40143a <explode_bomb>
   0x00000000004010cb <+105>:   nopl   0x0(%rax,%rax,1)
   0x00000000004010d0 <+110>:   jmp    0x4010d9 <phase_5+119>
   0x00000000004010d2 <+112>:   mov    $0x0,%eax   // eax=0
   0x00000000004010d7 <+117>:   jmp    0x40108b <phase_5+41> //跳41
   0x00000000004010d9 <+119>:   mov    0x18(%rsp),%rax
   0x00000000004010de <+124>:   xor    %fs:0x28,%rax
   0x00000000004010e7 <+133>:   je     0x4010ee <phase_5+140>
   0x00000000004010e9 <+135>:   call   0x400b30 <__stack_chk_fail@plt>
   0x00000000004010ee <+140>:   add    $0x20,%rsp
   0x00000000004010f2 <+144>:   pop    %rbx
   0x00000000004010f3 <+145>:   ret
```

这里进行了一步索引

也就是

text[i&0xf]==s[j]

```
(gdb) x/16bc 0x4024b0
0x4024b0 <array.3449>:  109 'm' 97 'a'  100 'd' 117 'u' 105 'i' 101 'e' 114 'r' 115 's'
0x4024b8 <array.3449+8>:        110 'n' 102 'f' 111 'o' 116 't' 118 'v' 98 'b'  121 'y' 108 'l'
```

可以得出一组字符串

)/.%&'

```
(gdb) run
Starting program: /mnt/c/Users/Retr0/Desktop/天问/第一周csapp/bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
1 311
Halfway there!
3 0
So you got that one.  Try this one.
)/.%&'
Good work!  On to the next...
```

第六题

```
   0x00000000004010f4 <+0>:     push   %r14
```

```
   0x00000000004010f6 <+2>:     push   %r13
   0x00000000004010f8 <+4>:     push   %r12
   0x00000000004010fa <+6>:     push   %rbp
   0x00000000004010fb <+7>:     push   %rbx
   0x00000000004010fc <+8>:     sub    $0x50,%rsp
   0x0000000000401100 <+12>:    mov    %rsp,%r13  // r13=rsp
   0x0000000000401103 <+15>:    mov    %rsp,%rsi //rsi=rsp
   0x0000000000401106 <+18>:    call   0x40145c <read_six_numbers> //read 6 nums
   0x000000000040110b <+23>:    mov    %rsp,%r14 // r14=rsp
   0x000000000040110e <+26>:    mov    $0x0,%r12d //r12d=0
   0x0000000000401114 <+32>:    mov    %r13,%rbp  //rbp=r13 = rsp
   0x0000000000401117 <+35>:    mov    0x0(%r13),%eax //eax =*(r13)
   0x000000000040111b <+39>:    sub    $0x1,%eax //eax-=1
   0x000000000040111e <+42>:    cmp    $0x5,%eax //eax cmp 5  //
   0x0000000000401121 <+45>:    jbe    0x401128 <phase_6+52> <= 5 跳转52
   0x0000000000401123 <+47>:    call   0x40143a <explode_bomb>
   0x0000000000401128 <+52>:    add    $0x1,%r12d  // r12d+=1
   0x000000000040112c <+56>:    cmp    $0x6,%r12d // cmp 6
   0x0000000000401130 <+60>:    je     0x401153 <phase_6+95> 等于跳转95
   0x0000000000401132 <+62>:    mov    %r12d,%ebx //ebx=r12d x/s
   0x0000000000401135 <+65>:    movslq %ebx,%rax  //rax=ebx
   0x0000000000401138 <+68>:    mov    (%rsp,%rax,4),%eax //eax=*(rsp+rax*4)
   0x000000000040113b <+71>:    cmp    %eax,0x0(%rbp)
   0x000000000040113e <+74>:    jne    0x401145 <phase_6+81> //*(rbp)和eax不相等跳
转
   0x0000000000401140 <+76>:    call   0x40143a <explode_bomb>
   0x0000000000401145 <+81>:    add    $0x1,%ebx //ebx+=1
   0x0000000000401148 <+84>:    cmp    $0x5,%ebx
   0x000000000040114b <+87>:    jle    0x401135 <phase_6+65> //ebx<=5 跳转
   0x000000000040114d <+89>:    add    $0x4,%r13
   0x0000000000401151 <+93>:    jmp    0x401114 <phase_6+32>
   0x0000000000401153 <+95>:    lea    0x18(%rsp),%rsi   // rsi=*(rsp+0x18)
   0x0000000000401158 <+100>:   mov    %r14,%rax //rax=r14
   0x000000000040115b <+103>:   mov    $0x7,%ecx //ecx=7
   0x0000000000401160 <+108>:   mov    %ecx,%edx //edx=ecx=7
   0x0000000000401162 <+110>:   sub    (%rax),%edx //edx-=*(rax)
   0x0000000000401164 <+112>:   mov    %edx,(%rax) //*(rax)=edx
   0x0000000000401166 <+114>:   add    $0x4,%rax rax+=4
   0x000000000040116a <+118>:   cmp    %rsi,%rax 比较rsi和rax
   0x000000000040116d <+121>:   jne    0x401160 <phase_6+108> // 不等跳转 108
   0x000000000040116f <+123>:   mov    $0x0,%esi //esi=0;
   0x0000000000401174 <+128>:   jmp    0x401197 <phase_6+163> //跳转
   0x0000000000401176 <+130>:   mov    0x8(%rdx),%rdx
   0x000000000040117a <+134>:   add    $0x1,%eax
   0x000000000040117d <+137>:   cmp    %ecx,%eax
   0x000000000040117f <+139>:   jne    0x401176 <phase_6+130>
   0x0000000000401181 <+141>:   jmp    0x401188 <phase_6+148>
   0x0000000000401183 <+143>:   mov    $0x6032d0,%edx   //edx=0x6032d0
   0x0000000000401188 <+148>:   mov    %rdx,0x20(%rsp,%rsi,2) //*
(rsp+rsi*2+0x20)=rdx
   0x000000000040118d <+153>:   add    $0x4,%rsi //rsi+=4
   0x0000000000401191 <+157>:   cmp    $0x18,%rsi
   0x0000000000401195 <+161>:   je     0x4011ab <phase_6+183> //rsi=0x18 跳
   0x0000000000401197 <+163>:   mov    (%rsp,%rsi,1),%ecx    //ecx=*(rsp+rsi)
   0x000000000040119a <+166>:   cmp    $0x1,%ecx  //
   0x000000000040119d <+169>:   jle    0x401183 <phase_6+143>  // ecx <=1 跳转143
   0x000000000040119f <+171>:   mov    $0x1,%eax
   0x00000000004011a4 <+176>:   mov    $0x6032d0,%edx
```

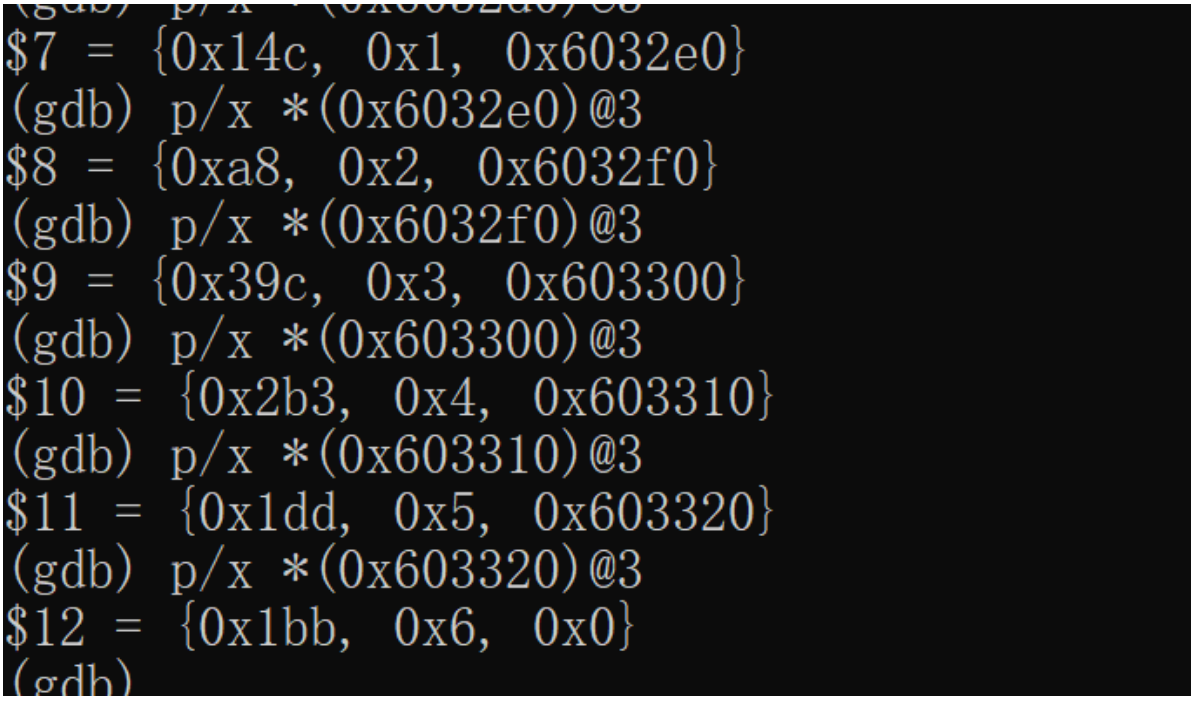
```
   0x00000000004011a9 <+181>:   jmp    0x401176 <phase_6+130>
   0x00000000004011ab <+183>:   mov    0x20(%rsp),%rbx   //rbx=*(rsp+0x20)
   0x00000000004011b0 <+188>:   lea    0x28(%rsp),%rax   //rax=*(rsp+0x28)
   0x00000000004011b5 <+193>:   lea    0x50(%rsp),%rsi //rsi=*(rsp+0x50)
   0x00000000004011ba <+198>:   mov    %rbx,%rcx //rcx=rbx
   0x00000000004011bd <+201>:   mov    (%rax),%rdx  //rdx=*(rax)
   0x00000000004011c0 <+204>:   mov    %rdx,0x8(%rcx) //*(rcx+0x8)=rdx
   0x00000000004011c4 <+208>:   add    $0x8,%rax //rax+=0x8
   0x00000000004011c8 <+212>:   cmp    %rsi,%rax
   0x00000000004011cb <+215>:   je     0x4011d2 <phase_6+222> 如果rax=rsi 跳
   0x00000000004011cd <+217>:   mov    %rdx,%rcx
   0x00000000004011d0 <+220>:   jmp    0x4011bd <phase_6+201>
   0x00000000004011d2 <+222>:   movq   $0x0,0x8(%rdx)  // *(rdx+0x8)=0
   0x00000000004011da <+230>:   mov    $0x5,%ebp //ebp=5
   0x00000000004011df <+235>:   mov    0x8(%rbx),%rax //rax=*(rbx+0x8)
   0x00000000004011e3 <+239>:   mov    (%rax),%eax //eax=*(rax)
   0x00000000004011e5 <+241>:   cmp    %eax,(%rbx)
   0x00000000004011e7 <+243>:   jge    0x4011ee <phase_6+250> //*(rbx)>=eax 跳转
   0x00000000004011e9 <+245>:   call   0x40143a <explode_bomb>
   0x00000000004011ee <+250>:   mov    0x8(%rbx),%rbx  // rbx=*(rbx+0x8)
   0x00000000004011f2 <+254>:   sub    $0x1,%ebp
   0x00000000004011f5 <+257>:   jne    0x4011df <phase_6+235> //ebp不等于1 跳转
   0x00000000004011f7 <+259>:   add    $0x50,%rsp
   0x00000000004011fb <+263>:   pop    %rbx
   0x00000000004011fc <+264>:   pop    %rbp
   0x00000000004011fd <+265>:   pop    %r12
   0x00000000004011ff <+267>:   pop    %r13
   0x0000000000401201 <+269>:   pop    %r14
```

这里前面先判断了 是否输入6个数字

然后每个数字 都要小于6

并且两两之间并不相同。

然后读一下地址位



```
0x00000000004011da <+230>:    mov    $0x5,%ebp //ebp=5
0x00000000004011df <+235>:    mov    0x8(%rbx),%rax //rax=*(rbx+0x8)
0x00000000004011e3 <+239>:    mov    (%rax),%eax //eax=*(rax)
0x00000000004011e5 <+241>:    cmp    %eax,(%rbx)
0x00000000004011e7 <+243>:    jge    0x4011ee <phase_6+250> //*(rbx)>=eax 跳转
```

按照这里可以看出是降序排列。

3 4 5 6 1 2

```
0x0000000000401158 <+100>:    mov    %r14,%rax //rax=r14
0x000000000040115b <+103>:    mov    $0x7,%ecx //ecx=7
0x0000000000401160 <+108>:    mov    %ecx,%edx //edx=ecx=7
0x0000000000401162 <+110>:    sub    (%rax),%edx //edx-=*(rax)
```

这里每个值得产生都是由7-去当前值得到的

所以可以得出答案序列

4 3 2 1 6 5

```
0x000000000040113b in phase_6 ()
(gdb) ni
0x000000000040113e in phase_6 ()
(gdb) ni
0x0000000000401145 in phase_6 ()
(gdb) ni
0x0000000000401148 in phase_6 ()
(gdb) ni
0x000000000040114b in phase_6 ()
(gdb) nin
Undefined command: "nin".  Try "help".
(gdb) ni
0x0000000000401135 in phase_6 ()
(gdb) ni a4
No symbol "a4" in current context.
(gdb) ni 0x4011c0

BOOM!!!
The bomb has blown up.
[Inferior 1 (process 194) exited with code 010]
(gdb) run
Starting program: /mnt/c/Users/RetrO/Desktop/天问/
Welcome to my fiendish little bomb. You have 6 pha
which to blow yourself up. Have a nice day!
Border relations with Canada have never been bett
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
1 311
Halfway there!

3 0
So you got that one.  Try this one.
)/.%&'
Good work!  On to the next...
4 3 2 1 6 5

Breakpoint 2, 0x00000000004010f4 in phase_6 ()
(gdb) continue
Continuing.
Congratulations! You've defused the bomb!
[Inferior 1 (process 195) exited normally]
```

secret_phase

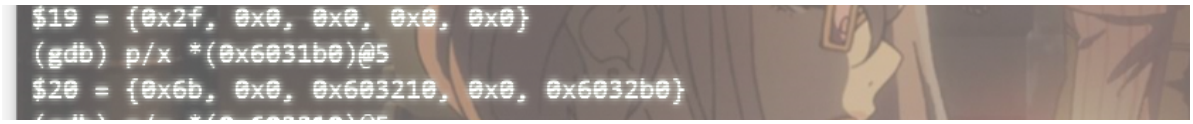
经过ida查看整个程序发现有一个secret_phase函数进入方式是在第四关输入正确值之后 加字符串DrEvil进入。

```
   0x0000000000401242 <+0>:     push   %rbx
   0x0000000000401243 <+1>:     call   0x40149e <read_line>
   0x0000000000401248 <+6>:     mov    $0xa,%edx //edx=10
   0x000000000040124d <+11>:    mov    $0x0,%esi //esi=0
   0x0000000000401252 <+16>:    mov    %rax,%rdi //rdi=rax
   0x0000000000401255 <+19>:    call   0x400bd0 <strtol@plt>
   0x000000000040125a <+24>:    mov    %rax,%rbx //rbx=rax
   0x000000000040125d <+27>:    lea    -0x1(%rax),%eax //eax=*(rax-0x1)
   0x0000000000401260 <+30>:    cmp    $0x3e8,%eax
   0x0000000000401265 <+35>:    jbe    0x40126c <secret_phase+42> // eax<=1000 跳
转
   0x0000000000401267 <+37>:    call   0x40143a <explode_bomb> //否则bomb了
   0x000000000040126c <+42>:    mov    %ebx,%esi    //esi=ebx
   0x000000000040126e <+44>:    mov    $0x6030f0,%edi //edi=0x6030f0
   0x0000000000401273 <+49>:    call   0x401204 <fun7> //call fun7
   0x0000000000401278 <+54>:    cmp    $0x2,%eax  //eax==2
   0x000000000040127b <+57>:    je     0x401282 <secret_phase+64> //相等跳转
   0x000000000040127d <+59>:    call   0x40143a <explode_bomb>  //否则bomb
   0x0000000000401282 <+64>:    mov    $0x402438,%edi //edi 一串字符串
   0x0000000000401287 <+69>:    call   0x400b10 <puts@plt> //puts出来
   0x000000000040128c <+74>:    call   0x4015c4 <phase_defused>
   0x0000000000401291 <+79>:    pop    %rbx
   0x0000000000401292 <+80>:    ret
```

func7

```
   0x0000000000401204 <+0>:     sub    $0x8,%rsp  //rsp+=8
   0x0000000000401208 <+4>:     test   %rdi,%rdi //rdi 不等于0跳转
   0x000000000040120b <+7>:     je     0x401238 <fun7+52>
   0x000000000040120d <+9>:     mov    (%rdi),%edx  //edx=*(rdi)
   0x000000000040120f <+11>:    cmp    %esi,%edx /
   0x0000000000401211 <+13>:    jle    0x401220 <fun7+28> // edx<=esi 跳转
   0x0000000000401213 <+15>:    mov    0x8(%rdi),%rdi // rdi=*(rdi+0x8)
   0x0000000000401217 <+19>:    call   0x401204 <fun7> //跳回去
   0x000000000040121c <+24>:    add    %eax,%eax //eax*=2
   0x000000000040121e <+26>:    jmp    0x40123d <fun7+57>  //跳57
   0x0000000000401220 <+28>:    mov    $0x0,%eax //eax=0
   0x0000000000401225 <+33>:    cmp    %esi,%edx
   0x0000000000401227 <+35>:    je     0x40123d <fun7+57> //edx==esi 跳转
   0x0000000000401229 <+37>:    mov    0x10(%rdi),%rdi //rdi=*(rdi+0x10)
   0x000000000040122d <+41>:    call   0x401204 <fun7> // 跳会去
   0x0000000000401232 <+46>:    lea    0x1(%rax,%rax,1),%eax //eax=*
(rax+rax+0x1)
   0x0000000000401236 <+50>:    jmp    0x40123d <fun7+57> //跳
   0x0000000000401238 <+52>:    mov    $0xffffffff,%eax //eax=2**32-1
   0x000000000040123d <+57>:    add    $0x8,%rsp  //rsp+=8
   0x0000000000401241 <+61>:    ret
```

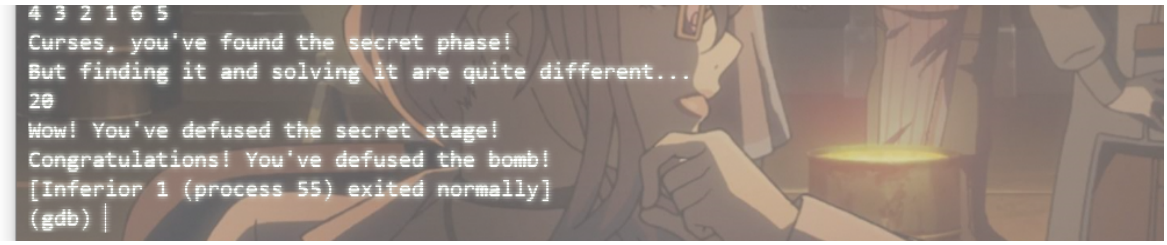
```
                36
        8                50
    6       22        45        107
1      7  20    35  40    47   99      1001
```



通过上面得方法可以得到上面得一个二叉树。

而我们0就是往左走

每次eax=*2 往右走是* *=2再+1 ， 所以我们只需要找20这个值应该就成功了。（好像必须是最底层结点才可以）



至此bomblab部分结束。。

再bomblab学到了很多。比如看汇编+看跳转 猜各种逻辑+ gdb一些基础的方法等等。