南京航空航天大学计算机科学与技术学院计算机组成原理实验 Bomblab 二进制炸弹实验报告

实验信息

姓名：李元成
学号：161630216
班级：161630216
指导教师：李博涵
实验日期：2018.6
我的炸弹编号：bomb3
我的答案：

```
For NASA, space is still a high priority.
8 9 26 61 148 357
1 -458
5 15 HappyHangHangXueZhang
jdoefg
3 2 6 4 5 1
50
```

第一关

考察内容

寻找地址的所存的字符串

反汇编代码分析

```
08048ae0 <phase_1>:
 8048ae0:  55                        push   %ebp
 8048ae1:  89 e5                     mov    %esp,%ebp
 8048ae3:  83 ec 18                  sub    $0x18,%esp
 8048ae6:  c7 44 24 04 80 a1 04      movl   $0x804a180,0x4(%esp)  答案
 8048aed:  08
 8048aee:  8b 45 08                  mov    0x8(%ebp),%eax
 8048af1:  89 04 24                  mov    %eax,(%esp)
 8048af4:  e8 99 04 00 00            call   8048f92 <strings_not_equal>
```

```
 8048af9:  85 c0                    test   %eax,%eax 比较，不同就炸
 8048afb:  74 05                    je     8048b02 <phase_1+0x22>
 8048afd:  e8 b3 06 00 00           call   80491b5 <explode_bomb>
 8048b02:  c9                       leave
 8048b03:  c3                       ret
```

操作过程或解题思路

```
(gdb) x/s 0x804a180
0x804a180:      "For NASA, space is still a high priority."
```

第二关

考察内容

for 循环

反汇编代码分析

```
08048b04 <phase_2>:
 8048b04:  55                       push   %ebp
 8048b05:  89 e5                    mov    %esp,%ebp
 8048b07:  56                       push   %esi
 8048b08:  53                       push   %ebx
 8048b09:  83 ec 30                 sub    $0x30,%esp
 8048b0c:  8d 45 e0                 lea    -0x20(%ebp),%eax
 8048b0f:  89 44 24 04              mov    %eax,0x4(%esp)
 8048b13:  8b 45 08                 mov    0x8(%ebp),%eax
 8048b16:  89 04 24                 mov    %eax,(%esp)
 8048b19:  e8 d9 06 00 00           call    80491f7 <read_six_numbers>
 读数
 8048b1e:  8b 45 e0                 mov    -0x20(%ebp),%eax
 8048b21:  83 f8 07                 cmp    $0x7,%eax 第一个大于 7 就炸
 8048b24:  7e 05                    jle    8048b2b <phase_2+0x27>
 8048b26:  3b 45 e4                 cmp    -0x1c(%ebp),%eax 第二个小于
第一个就炸
 8048b29:  7e 22                    jle    8048b4d <phase_2+0x49>
 8048b2b:  e8 85 06 00 00           call   80491b5 <explode_bomb>
```

```
8048b30:  eb 1b                    jmp    8048b4d <phase_2+0x49>
8048b32:  8b 53 fc                 mov    -0x4(%ebx),%edx edx=ebx-4
8048b35:  8b 43 f8                 mov    -0x8(%ebx),%eax eax=ebx-8
8048b38:  8d 04 50                 lea    (%eax,%edx,2),%eax
eax+=edx*2
8048b3b:  39 03                    cmp    %eax,(%ebx)   eax 和 ebx 不等
就炸
8048b3d:  74 05                    je     8048b44 <phase_2+0x40>
8048b3f:  e8 71 06 00 00           call   80491b5 <explode_bomb>
8048b44:  83 c3 04                 add    $0x4,%ebx ebx+=4
8048b47:  39 f3                    cmp    %esi,%ebx for 循环控制
8048b49:  75 e7                    jne    8048b32 <phase_2+0x2e>
8048b4b:  eb 08                    jmp    8048b55 <phase_2+0x51>
8048b4d:  8d 5d e8                 lea    -0x18(%ebp),%ebx
8048b50:  8d 75 f8                 lea    -0x8(%ebp),%esi
8048b53:  eb dd                    jmp    8048b32 <phase_2+0x2e>
8048b55:  83 c4 30                 add    $0x30,%esp
8048b58:  5b                       pop    %ebx
8048b59:  5e                       pop    %esi
8048b5a:  5d                       pop    %ebp
8048b5b:  c3                       ret
```

操作过程或解题思路

```
(gdb) u *0x08048b38
0x08048b38 in phase_2 ()
(gdb) p $edx
$2 = 9
(gdb) p $eax
$3 = 8
```

eax 放的是第一个数。edx 是第二个
eax=eax + edx*2
而 ebx 放的是第三个数
所以 f(n)=f(n-1)*2+f(n-2)
我前两个是 8 9
所以 8 9 26 61 148 357

第三关

考察内容

Switch 语句

反汇编代码分析

```
08048b5c <phase_3>:
 8048b5c:  55                          push   %ebp
 8048b5d:  89 e5                       mov    %esp,%ebp
 8048b5f:  83 ec 28                    sub    $0x28,%esp
 8048b62:  8d 45 f0                    lea    -0x10(%ebp),%eax
 8048b65:  89 44 24 0c                 mov    %eax,0xc(%esp)
 8048b69:  8d 45 f4                    lea    -0xc(%ebp),%eax
 8048b6c:  89 44 24 08                 mov    %eax,0x8(%esp)
 8048b70:  c7 44 24 04 dd a3 04        movl   $0x804a3dd,0x4(%esp)  %d %d
 8048b77:  08
 8048b78:  8b 45 08                    mov    0x8(%ebp),%eax
 8048b7b:  89 04 24                    mov    %eax,(%esp)
 8048b7e:  e8 5d fc ff ff              call                          80487e0
<__isoc99_sscanf@plt>                                                 读数
 8048b83:  83 f8 01                    cmp    $0x1,%eax  输入个数小于 2 就
炸
 8048b86:  7f 05                       jg     8048b8d <phase_3+0x31>
 8048b88:  e8 28 06 00 00              call   80491b5 <explode_bomb>
 8048b8d:  83 7d f4 07                 cmpl    $0x7,-0xc(%ebp)  如果第一个
数大于 7 就炸
 8048b91:  77 65                       ja     8048bf8 <phase_3+0x9c>
 8048b93:  8b 45 f4                    mov    -0xc(%ebp),%eax
 8048b96:  ff 24 85 dc a1 04 08        jmp    *0x804a1dc(,%eax,4)
 8048b9d:  b8 00 00 00 00              mov    $0x0,%eax
 8048ba2:  eb 05                       jmp    8048ba9 <phase_3+0x4d>
 8048ba4:  b8 2b 01 00 00              mov    $0x12b,%eax
 8048ba9:  2d 2b 02 00 00              sub    $0x22b,%eax
 8048bae:  eb 05                       jmp    8048bb5 <phase_3+0x59>
 8048bb0:  b8 00 00 00 00              mov    $0x0,%eax
 8048bb5:  05 c7 02 00 00              add    $0x2c7,%eax
 8048bba:  eb 05                       jmp    8048bc1 <phase_3+0x65>
 8048bbc:  b8 00 00 00 00              mov    $0x0,%eax
 8048bc1:  2d 66 02 00 00              sub    $0x266,%eax
 8048bc6:  eb 05                       jmp    8048bcd <phase_3+0x71>
 8048bc8:  b8 00 00 00 00              mov    $0x0,%eax
 8048bcd:  05 66 02 00 00              add    $0x266,%eax
 8048bd2:  eb 05                       jmp    8048bd9 <phase_3+0x7d>
 8048bd4:  b8 00 00 00 00              mov    $0x0,%eax
 8048bd9:  2d 66 02 00 00              sub    $0x266,%eax
 8048bde:  eb 05                       jmp    8048be5 <phase_3+0x89>
 8048be0:  b8 00 00 00 00              mov    $0x0,%eax
```

```
 8048be5:  05 66 02 00 00          add    $0x266,%eax
 8048bea:  eb 05                   jmp    8048bf1 <phase_3+0x95>
 8048bec:  b8 00 00 00 00          mov    $0x0,%eax
 8048bf1:  2d 66 02 00 00          sub    $0x266,%eax
 8048bf6:  eb 0a                   jmp    8048c02 <phase_3+0xa6>
 8048bf8:  e8 b8 05 00 00          call   80491b5 <explode_bomb>
 8048bfd:  b8 00 00 00 00          mov    $0x0,%eax
 8048c02:  83 7d f4 05             cmpl   $0x5,-0xc(%ebp)
 8048c06:  7f 05                   jg     8048c0d <phase_3+0xb1>
 8048c08:  3b 45 f0                cmp    -0x10(%ebp),%eax 第二个数和
各种操作后的第一个数相比，不等就炸
 8048c0b:  74 05                   je     8048c12 <phase_3+0xb6>
 8048c0d:  e8 a3 05 00 00          call   80491b5 <explode_bomb>
 8048c12:  c9                      leave
 8048c13:  c3                      ret
```

操作过程或解题思路

```
(gdb) x/d $ebp-0x10
0xffffd5e8:     -458
```

答案是

1 -458

第四关

考察内容

递归

反汇编代码分析

```
08048c72 <phase_4>:
 8048c72:  55                      push   %ebp
 8048c73:  89 e5                   mov    %esp,%ebp
 8048c75:  83 ec 28                sub    $0x28,%esp
 8048c78:  8d 45 f0                lea    -0x10(%ebp),%eax
 8048c7b:  89 44 24 0c             mov    %eax,0xc(%esp)
 8048c7f:  8d 45 f4                lea    -0xc(%ebp),%eax
 8048c82:  89 44 24 08             mov    %eax,0x8(%esp)
 8048c86:  c7 44 24 04 dd a3 04    movl   $0x804a3dd,0x4(%esp)
```

```
 8048c8d:  08
 8048c8e:  8b 45 08                mov    0x8(%ebp),%eax
 8048c91:  89 04 24                mov    %eax,(%esp)
 8048c94:  e8 47 fb ff ff          call                        80487e0
<__isoc99_sscanf@plt>     和 phase_3 一样的套路
 8048c99:  83 f8 02                cmp    $0x2,%eax  个数不是 2 就炸
 8048c9c:  75 06                   jne    8048ca4 <phase_4+0x32>
 8048c9e:  83 7d f4 0e             cmpl    $0xe,-0xc(%ebp)  第一个数小
于 0xe，不然就炸
 8048ca2:  76 05                   jbe    8048ca9 <phase_4+0x37>
 8048ca4:  e8 0c 05 00 00          call   80491b5 <explode_bomb>
 8048ca9:  c7 44 24 08 0e 00 00    movl   $0xe,0x8(%esp)准备参数
 8048cb0:  00
 8048cb1:  c7 44 24 04 00 00 00    movl   $0x0,0x4(%esp)  准备参数
 8048cb8:  00
 8048cb9:  8b 45 f4                mov    -0xc(%ebp),%eax
 8048cbc:  89 04 24                mov    %eax,(%esp)  准备参数
 8048cbf:  e8 50 ff ff ff          call   8048c14 <func4>调用 func4
 8048cc4:  83 f8 0f                cmp    $0xf,%eax  返回值不是 15 就炸
 8048cc7:  75 06                   jne    8048ccf <phase_4+0x5d>
 8048cc9:  83 7d f0 0f             cmpl    $0xf,-0x10(%ebp)第二个数不
是 15 也要炸
 8048ccd:  74 05                   je     8048cd4 <phase_4+0x62>
 8048ccf:  e8 e1 04 00 00          call   80491b5 <explode_bomb>
 8048cd4:  c9                      leave
 8048cd5:  c3                      ret


08048c14 <func4>:
 8048c14:  55                      push   %ebp
 8048c15:  89 e5                   mov    %esp,%ebp
 8048c17:  56                      push   %esi
 8048c18:  53                      push   %ebx
 8048c19:  83 ec 10                sub    $0x10,%esp
 8048c1c:  8b 55 08                mov    0x8(%ebp),%edx  取出参数
 8048c1f:  8b 45 0c                mov    0xc(%ebp),%eax 取出参数
 8048c22:  8b 75 10                mov    0x10(%ebp),%esi 取出参数
 8048c25:  89 f1                   mov    %esi,%ecx
 8048c27:  29 c1                   sub    %eax,%ecx  ecx-=eax
 8048c29:  89 cb                   mov    %ecx,%ebx
 8048c2b:  c1 eb 1f                shr    $0x1f,%ebx  ebx>>31,取零
 8048c2e:  01 d9                   add    %ebx,%ecx  ecx+=ebx
 8048c30:  d1 f9                   sar    %ecx  ecx/=2
 8048c32:  8d 1c 01                lea    (%ecx,%eax,1),%ebx
```

```
ebx=ecx+eax
8048c35:  39 d3                   cmp    %edx,%ebx 如果 edx<=ebx 就跳
8048c37:  7e 17                   jle    8048c50 <func4+0x3c>
8048c39:  8d 4b ff                lea    -0x1(%ebx),%ecx 准备参数
8048c3c:  89 4c 24 08             mov    %ecx,0x8(%esp) ebx-1
8048c40:  89 44 24 04             mov    %eax,0x4(%esp) eax
8048c44:  89 14 24                mov    %edx,(%esp)edx
8048c47:  e8 c8 ff ff ff          call   8048c14 <func4> 调用 func4
8048c4c:  01 d8                   add    %ebx,%eax ebx+=func4
8048c4e:  eb 1b                   jmp    8048c6b <func4+0x57> return
8048c50:  89 d8                   mov    %ebx,%eax else   eax=ebx
8048c52:  39 d3                   cmp    %edx,%ebx if(edx<=ebx)就跳
8048c54:  7d 15                   jge    8048c6b <func4+0x57> return
ebx
8048c56:  89 74 24 08             mov    %esi,0x8(%esp) 准备参数 esi
8048c5a:  8d 43 01                lea    0x1(%ebx),%eax ebx++
8048c5d:  89 44 24 04             mov    %eax,0x4(%esp) eax
8048c61:  89 14 24                mov    %edx,(%esp) edx
8048c64:  e8 ab ff ff ff          call   8048c14 <func4> 调用 func4
8048c69:  01 d8                   add    %ebx,%eax ebx+=func4
8048c6b:  83 c4 10                add    $0x10,%esp return ebx
8048c6e:  5b                      pop    %ebx
8048c6f:  5e                      pop    %esi
8048c70:  5d                      pop    %ebp
8048c71:  c3                      ret
```

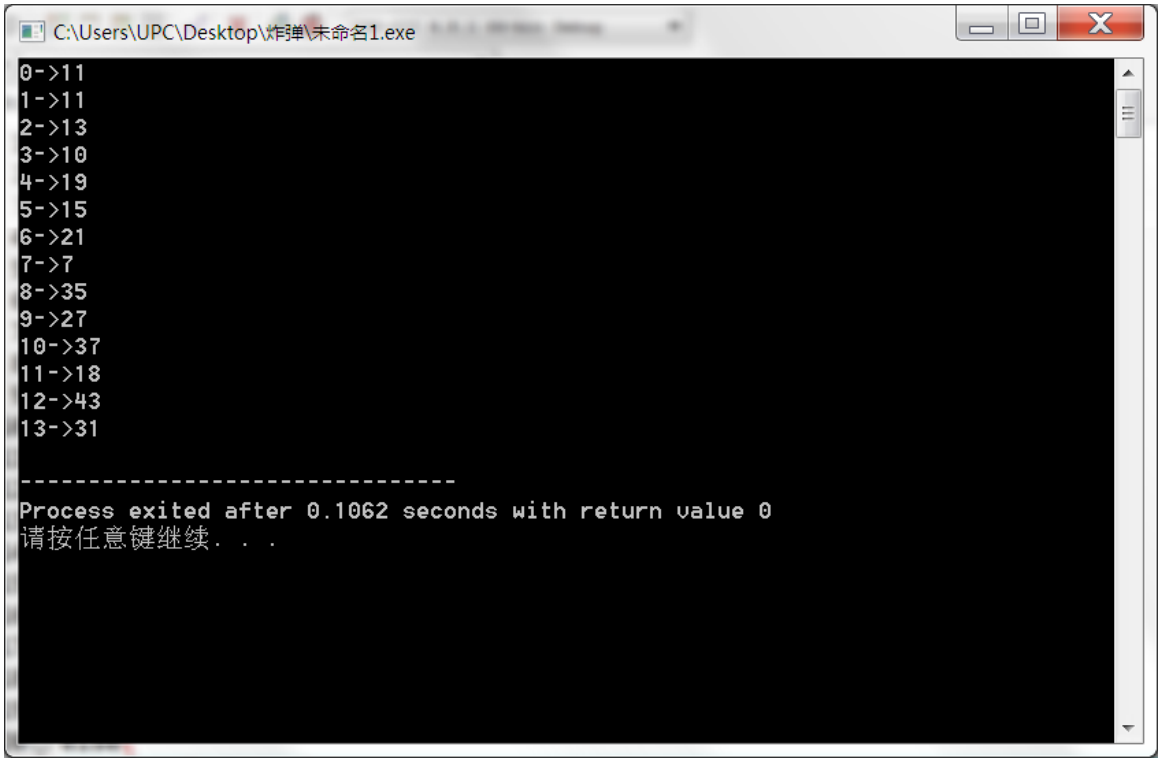
操作过程或解题思路

很明显，第二个数是 15
难点是第一个数输进去多少，写个 func4 程序轻松解决

```
int func(int x,int y,int z)
{int k;
int zz=z-y;
k=0;
zz/=2;
k=zz+y;
if(k>x)
{
y=k+func(x,y,k-1); //func(5,0,6)
return y;
}
else
{
```

```
y=k;
if(x<=k)
{
return y;
}
else{
y=k+func(x,k+1,z);

}
}
}
```



第一个是 5

所以答案是 5 15

第五关

考察内容

数组

反汇编代码分析

```
08048cd6 <phase_5>:
 8048cd6:  55                          push   %ebp
 8048cd7:  89 e5                       mov    %esp,%ebp
 8048cd9:  53                          push   %ebx
 8048cda:  83 ec 24                    sub    $0x24,%esp
 8048cdd:  8b 5d 08                    mov    0x8(%ebp),%ebx
 8048ce0:  89 1c 24                    mov    %ebx,(%esp)
 8048ce3:  e8 88 02 00 00              call   8048f70 <string_length>
 8048ce8:  83 f8 06                    cmp    $0x6,%eax 字符串长度不是 6
就炸
 8048ceb:  74 45                       je     8048d32 <phase_5+0x5c>
 8048ced:  e8 c3 04 00 00              call   80491b5 <explode_bomb>
 8048cf2:  eb 3e                       jmp    8048d32 <phase_5+0x5c>接着
跳
 8048cf4:  0f b6 14 03                 movzbl (%ebx,%eax,1),%edx
 edx=ebx+eax
 8048cf8:  83 e2 0f                    and    $0xf,%edx
 8048cfb:  0f b6 92 fc a1 04 08        movzbl 0x804a1fc(%edx),%edx
 8048d02:  88 54 05 f1                 mov    %dl,-0xf(%ebp,%eax,1)
 8048d06:  83 c0 01                    add    $0x1,%eax
 8048d09:  83 f8 06                    cmp    $0x6,%eax eax 控制循环
 8048d0c:  75 e6                       jne    8048cf4 <phase_5+0x1e> 循环
 结束跳到这里
 8048d0e:  c6 45 f7 00                 movb   $0x0,-0x9(%ebp)
 8048d12:  c7 44 24 04 d2 a1 04        movl   $0x804a1d2,0x4(%esp) 答案
 8048d19:  08
 8048d1a:  8d 45 f1                    lea    -0xf(%ebp),%eax 输入字符串
被转变后放在 ebp-0xf
 8048d1d:  89 04 24                    mov    %eax,(%esp)
 8048d20:  e8 6d 02 00 00              call   8048f92 <strings_not_equal>
 8048d25:  85 c0                       test   %eax,%eax 相比较，不等就炸
 8048d27:  74 10                       je     8048d39 <phase_5+0x63>
 8048d29:  e8 87 04 00 00              call   80491b5 <explode_bomb>
 8048d2e:  66 90                       xchg   %ax,%ax
 8048d30:  eb 07                       jmp    8048d39 <phase_5+0x63>
 8048d32:  b8 00 00 00 00              mov    $0x0,%eax eax=0
 8048d37:  eb bb                       jmp    8048cf4 <phase_5+0x1e>
 8048d39:  83 c4 24                    add    $0x24,%esp
 8048d3c:  5b                          pop    %ebx
 8048d3d:  5d                          pop    %ebp
 8048d3e:  c3                          ret
```

操作过程或解题思路

```
(gdb) x/s 0x804a1d2
0x804a1d2:      "oilers"
```

答案是 oilers

很明显，对于一个字母，它被转变后的字母是唯一的，挨个试就好了

```
phase_5 码表
e->e
o->l
i->f
l->v
r->d
q->a
s->u
d->i
f->r
g->s
oilers
jdoefg
```

第六关

考察内容

链表

反汇编代码分析

```
08048d3f <phase_6>:
 8048d3f:  55                          push   %ebp
 8048d40:  89 e5                       mov    %esp,%ebp
 8048d42:  56                          push   %esi
 8048d43:  53                          push   %ebx
 8048d44:  83 ec 40                    sub    $0x40,%esp
 8048d47:  8d 45 e0                    lea    -0x20(%ebp),%eax
 8048d4a:  89 44 24 04                 mov    %eax,0x4(%esp)
 8048d4e:  8b 45 08                    mov    0x8(%ebp),%eax
 8048d51:  89 04 24                    mov    %eax,(%esp)
 8048d54:  e8 9e 04 00 00              call    80491f7 <read_six_numbers>
```

读 6 个数

```
 8048d59:  be 00 00 00 00          mov    $0x0,%esi  esi 置 0
 8048d5e:  8b 44 b5 e0             mov    -0x20(%ebp,%esi,4),%eax
 8048d62:  83 e8 01                sub    $0x1,%eax
 8048d65:  83 f8 05                cmp    $0x5,%eax  大于 5
8048d68:   76 05                   jbe    8048d6f <phase_6+0x30>
 8048d6a:  e8 46 04 00 00          call   80491b5 <explode_bomb>
 8048d6f:  83 c6 01                add    $0x1,%esi
 8048d72:  83 fe 06                cmp    $0x6,%esi
 8048d75:  74 1b                   je     8048d92 <phase_6+0x53>
 8048d77:  89 f3                   mov    %esi,%ebx
 8048d79:  8b 44 9d e0             mov    -0x20(%ebp,%ebx,4),%eax
 8048d7d:  39 44 b5 dc             cmp    %eax,-0x24(%ebp,%esi,4)
 8048d81:  75 05                   jne    8048d88 <phase_6+0x49>
 8048d83:  e8 2d 04 00 00          call   80491b5 <explode_bomb>
 8048d88:  83 c3 01                add    $0x1,%ebx
 8048d8b:  83 fb 05                cmp    $0x5,%ebx
 8048d8e:  7e e9                   jle    8048d79 <phase_6+0x3a>
 8048d90:  eb cc                   jmp    8048d5e <phase_6+0x1f>
 8048d92:  8d 45 e0                lea    -0x20(%ebp),%eax
 8048d95:  8d 5d f8                lea    -0x8(%ebp),%ebx
 8048d98:  b9 07 00 00 00          mov    $0x7,%ecx
 8048d9d:  89 ca                   mov    %ecx,%edx
 8048d9f:  2b 10                   sub    (%eax),%edx
 8048da1:  89 10                   mov    %edx,(%eax)
 8048da3:  83 c0 04                add    $0x4,%eax
 8048da6:  39 d8                   cmp    %ebx,%eax
 8048da8:  75 f3                   jne    8048d9d <phase_6+0x5e>
 8048daa:  bb 00 00 00 00          mov    $0x0,%ebx
 8048daf:  eb 1d                   jmp    8048dce <phase_6+0x8f>
 8048db1:  8b 52 08                mov    0x8(%edx),%edx
 8048db4:  83 c0 01                add    $0x1,%eax
 8048db7:  39 c8                   cmp    %ecx,%eax
 8048db9:  75 f6                   jne    8048db1 <phase_6+0x72>
 8048dbb:  eb 05                   jmp    8048dc2 <phase_6+0x83>
 8048dbd:  ba 54 c1 04 08          mov    $0x804c154,%edx
 8048dc2:  89 54 b5 c8             mov    %edx,-0x38(%ebp,%esi,4)
 8048dc6:  83 c3 01                add    $0x1,%ebx
 8048dc9:  83 fb 06                cmp    $0x6,%ebx
 8048dcc:  74 17                   je     8048de5 <phase_6+0xa6>
 8048dce:  89 de                   mov    %ebx,%esi
 8048dd0:  8b 4c 9d e0             mov    -0x20(%ebp,%ebx,4),%ecx
 8048dd4:  83 f9 01                cmp    $0x1,%ecx
 8048dd7:  7e e4                   jle    8048dbd <phase_6+0x7e>
 8048dd9:  b8 01 00 00 00          mov    $0x1,%eax
```

```
8048dde:  ba 54 c1 04 08          mov    $0x804c154,%edx  链表的头
8048de3:  eb cc                   jmp    8048db1 <phase_6+0x72>
8048de5:  8b 5d c8                mov    -0x38(%ebp),%ebx
8048de8:  8d 45 cc                lea    -0x34(%ebp),%eax
8048deb:  8d 75 e0                lea    -0x20(%ebp),%esi
8048dee:  89 d9                   mov    %ebx,%ecx
8048df0:  8b 10                   mov    (%eax),%edx
8048df2:  89 51 08                mov    %edx,0x8(%ecx)
8048df5:  83 c0 04                add    $0x4,%eax
8048df8:  39 f0                   cmp    %esi,%eax  循环
8048dfa:  74 04                   je     8048e00 <phase_6+0xc1>
8048dfc:  89 d1                   mov    %edx,%ecx
8048dfe:  eb f0                   jmp    8048df0 <phase_6+0xb1>
8048e00:  c7 42 08 00 00 00 00    movl   $0x0,0x8(%edx)
8048e07:  be 05 00 00 00          mov    $0x5,%esi
8048e0c:  8b 43 08                mov    0x8(%ebx),%eax
8048e0f:  8b 00                   mov    (%eax),%eax
8048e11:  39 03                   cmp    %eax,(%ebx)  前后项对比，不是
后>前就炸
8048e13:  7d 05                   jge    8048e1a <phase_6+0xdb>
8048e15:  e8 9b 03 00 00          call   80491b5 <explode_bomb>
8048e1a:  8b 5b 08                mov    0x8(%ebx),%ebx
8048e1d:  83 ee 01                sub    $0x1,%esi
8048e20:  75 ea                   jne    8048e0c <phase_6+0xcd>
8048e22:  83 c4 40                add    $0x40,%esp
8048e25:  5b                      pop    %ebx
8048e26:  5e                      pop    %esi
8048e27:  5d                      pop    %ebp
8048e28:  c3                      ret
```

操作过程或解题思路

其实中间大段的 for 循环我是没有看的，因为是没有必要的。重点是这个链表头

```
(gdb) x/d 0x804c154
0x804c154 <node1>:      624
```

```
(gdb) x/d 0x804c154+0x4
0x804c158 <node1+4>:    1
(gdb) x/d 0x804c154+0xc
0x804c160 <node2>:      369
(gdb) x/d 0x804c154+0xc+0x4
0x804c164 <node2+4>:    2
```

可以推断出这个链表

```
struct node{
int x;
int no;
node *next;
};
```

x 对于当前地址，no 对应+0x4，*next 对应 0x8
x 和 no 一一对应，对于每个 no，都有唯一的 x 相对应。
最后对 x 进行倒叙检查，后面节点的 x 要小于前面的
现在的任务是排序，先写出整个对应关系
1->624
2->369
3->472
4->977
5->656
6->359
排序结果为
节点的顺序也和输入数的顺序有关系
1 2 3 4 5 6 输入
4 5 6 1 2 3 映射结束

所以答案是
3 2 6 4 5 1

隐藏关

考察内容

二叉搜索树

进入本关的方法

```
08048e29 <fun7>:
```

（其实是看大佬日了 7 个才知道要进入隐藏关）
首先在 phase_defused

这个说明可能是第三和第四关再输一个特殊的字符串

```
(gdb) x/s 0x804c8f0
0x804c8f0 <input_strings+240>:  "5 15
```

很明显，这是第四关的数字

```
(gdb) x/s 0x804a440
0x804a440:      "HappyHangHangXueZhang"
```

密码已给出，再四关输入个字符串即可

```
(gdb) x/s 0x804a437
0x804a437:      "%d %d %s"
```

反汇编代码分析

```
08048e7c <secret_phase>:
 8048e7c:  55                          push   %ebp
 8048e7d:  89 e5                       mov    %esp,%ebp
 8048e7f:  53                          push   %ebx
 8048e80:  83 ec 14                    sub    $0x14,%esp
 8048e83:  e8 be 03 00 00              call    8049246 <read_line> 读一个
数
 8048e88:  c7 44 24 08 0a 00 00        movl   $0xa,0x8(%esp)
 8048e8f:  00
 8048e90:  c7 44 24 04 00 00 00        movl   $0x0,0x4(%esp)
 8048e97:  00
 8048e98:  89 04 24                    mov    %eax,(%esp)
 8048e9b:  e8 a0 f9 ff ff              call   8048840 <strtol@plt>
 8048ea0:  89 c3                       mov    %eax,%ebx eax 是我们输入的数
 8048ea2:  8d 40 ff                    lea    -0x1(%eax),%eax  eax--
 8048ea5:  3d e8 03 00 00              cmp    $0x3e8,%eax eax 要比 0x3e8 小
 8048eaa:  76 05                       jbe    8048eb1 <secret_phase+0x35>
 8048eac:  e8 04 03 00 00              call   80491b5 <explode_bomb>
 8048eb1:  89 5c 24 04                 mov    %ebx,0x4(%esp)  准备参数
 8048eb5:  c7 04 24 a0 c0 04 08        movl    $0x804c0a0,(%esp)  这个地址
放的是 36
 8048ebc:  e8 68 ff ff ff              call   8048e29 <fun7>
 8048ec1:  83 f8 01                    cmp    $0x1,%eax  返回值和 1 比较
 8048ec4:  74 05                       je     8048ecb <secret_phase+0x4f>
 8048ec6:  e8 ea 02 00 00              call   80491b5 <explode_bomb>
 8048ecb:  c7 04 24 ac a1 04 08        movl   $0x804a1ac,(%esp)
 8048ed2:  e8 a9 f8 ff ff              call   8048780 <puts@plt>
 8048ed7:  e8 a2 04 00 00              call   804937e <phase_defused>
 8048edc:  83 c4 14                    add    $0x14,%esp
 8048edf:  5b                          pop    %ebx
 8048ee0:  5d                          pop    %ebp
 8048ee1:  c3                          ret
 8048ee2:  66 90                       xchg   %ax,%ax
```

```
8048ee4:  66 90                   xchg   %ax,%ax
8048ee6:  66 90                   xchg   %ax,%ax
8048ee8:  66 90                   xchg   %ax,%ax
8048eea:  66 90                   xchg   %ax,%ax
8048eec:  66 90                   xchg   %ax,%ax
8048eee:  66 90                   xchg   %ax,%ax

08048e29 <fun7>:
8048e29:  55                      push   %ebp
8048e2a:  89 e5                   mov    %esp,%ebp
8048e2c:  53                      push   %ebx
8048e2d:  83 ec 14                sub    $0x14,%esp
8048e30:  8b 55 08                mov    0x8(%ebp),%edx
8048e33:  8b 4d 0c                mov    0xc(%ebp),%ecx
8048e36:  85 d2                   test   %edx,%edx 放的$符号，自与，
zf=0，je 不会跳
8048e38:  74 37                   je     8048e71 <fun7+0x48>
8048e3a:  8b 1a                   mov    (%edx),%ebx $码是 36
8048e3c:  39 cb                   cmp    %ecx,%ebx 我们输入的数和 36
比
8048e3e:  7e 13                   jle    8048e53 <fun7+0x2a>大于等于
就跳
8048e40:  89 4c 24 04             mov    %ecx,0x4(%esp)
8048e44:  8b 42 04                mov    0x4(%edx),%eax
8048e47:  89 04 24                mov    %eax,(%esp)
8048e4a:  e8 da ff ff ff          call   8048e29 <fun7>
8048e4f:  01 c0                   add    %eax,%eax
8048e51:  eb 23                   jmp    8048e76 <fun7+0x4d>
8048e53:  b8 00 00 00 00          mov    $0x0,%eax eax 置零
8048e58:  39 cb                   cmp    %ecx,%ebx 等于就跳
8048e5a:  74 1a                   je     8048e76 <fun7+0x4d>
8048e5c:  89 4c 24 04             mov    %ecx,0x4(%esp)准备参数 50
8048e60:  8b 42 08                mov    0x8(%edx),%eax
8048e63:  89 04 24                mov    %eax,(%esp)  “2”码为 50
8048e66:  e8 be ff ff ff          call   8048e29 <fun7>
8048e6b:  8d 44 00 01             lea    0x1(%eax,%eax,1),%eax
8048e6f:  eb 05                   jmp    8048e76 <fun7+0x4d>
8048e71:  b8 ff ff ff ff          mov    $0xffffffff,%eax
8048e76:  83 c4 14                add    $0x14,%esp
8048e79:  5b                      pop    %ebx
8048e7a:  5d                      pop    %ebp
8048e7b:  c3                      ret
```

操作过程或解题思路

fun7（int x,char c）
x 是要寻找的数，c 是 ascii 是左边或者右边的数
左边是 36，右边是 50
返回值可能是 bool 0~1 表示找没找到
也有可能是寻找的次数
所以我先输了 36，发现 secret_phase 的 eax 是 0
所以，答案是 50

思考与体会

对汇编代码和 gdb 调试程序更加熟练。
遇到不会的可以跳过，把它当成黑箱，不用关心它的原理，只关心它的输入和输出，再通过输入输出寻找关系，逆推实现原理。
cmp 附近有 bomb 函数，突破点往往在这里。
回忆起大一上也做过这个实验，自己运气好蒙对了 5 关。这次很顺畅的通完 7 关，我变强了。

6	bomb3	Sun Jun 10 16:49	7	16	69	valid