- 姓名：傅锦龙
- 班级：162130
- 学号：162130117
- 报告阶段：lab2
- 完成日期：2023.5.26
- 本次实验，我完成了所有内容。

目录

1. phase_1

- 思路

```
08048ae0 <phase_1>:
 8048ae0:   83 ec 1c                sub    $0x1c,%esp
 8048ae3:   c7 44 24 04 08 a2 04    movl   $0x804a208,0x4(%esp)
 8048aea:   08
 8048aeb:   8b 44 24 20             mov    0x20(%esp),%eax
 8048aef:   89 04 24                mov    %eax,(%esp)
 8048af2:   e8 e3 04 00 00          call   8048fda <strings_not_equal>
 8048af7:   85 c0                   test   %eax,%eax
 8048af9:   74 05                   je     8048b00 <phase_1+0x20>
 8048afb:   e8 ef 06 00 00          call   80491ef <explode_bomb>
 8048b00:   83 c4 1c                add    $0x1c,%esp
 8048b03:   c3                      ret
```

- 第3行，为函数strings_not_equal传入第二个参数，即0x804a208，是一个地址
- 第5、6行，为函数strings_not_equal传入第一个参数，即我们输入的字符串
- 第7行，调用函数strings_not_equal返回值储存在%eax中，判断其是否为0，若为0，则跳至第11行，炸弹拆除成功，否则就会爆炸

于是，只需利用x/s指令查看0x804a208位置对应内存存的字符串即可：

```
(gdb) x/s 0x804a208
0x804a208:      "He is evil and fits easily into most overhead storage bins."
```

所以答案是：

```
He is evil and fits easily into most overhead storage bins.
```

- 完成截图

```
fujinlong@ubuntu:/mnt/hgfs/course/bomb42$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
```

2. phase_2

- 思路

phase_2首先通过read_six_numbers函数将输入的6个数字存储到以0x18(%esp)为首地址的连续内存中，而剩余phase_2等价C语言代码如下：

```
// 检查第一个数字是否为1
if (arr[0] != 1) {
    explode_bomb();
}

// 检查相邻两个数字的乘积是否为前一个数字的两倍
for (i = 1; i < 6; i++) {
    val = arr[i-1] * 2;
    if (val != arr[i]) {
        explode_bomb();
    }
}
```

由此可知答案为：

```
1 2 4 8 16 32
```

- 完成截图

```
1 2 4 8 16 32
That's number 2.  Keep going!
```

3. phase_3

- 思路

```
8048b55:    8d 44 24 28             lea    0x28(%esp),%eax
8048b59:    89 44 24 10             mov    %eax,0x10(%esp)
8048b5d:    8d 44 24 27             lea    0x27(%esp),%eax
8048b61:    89 44 24 0c             mov    %eax,0xc(%esp)
8048b65:    8d 44 24 2c             lea    0x2c(%esp),%eax
8048b69:    89 44 24 08             mov    %eax,0x8(%esp)
8048b6d:    c7 44 24 04 6a a2 04    movl   $0x804a26a,0x4(%esp)
```

```
(gdb) x/s 0x804a26a
0x804a26a:      "%d %c %d"
```

通过查看0x804a26a，可知需要3个输入，且依次位于0x2c(%esp)，0x27(%esp)，0x28(%esp)，设为a,b,c

```
8048b8b:    83 7c 24 2c 07          cmpl   $0x7,0x2c(%esp)
8048b90:    0f 87 f2 00 00 00       ja     8048c88 <phase_3+0x136>
8048b96:    8b 44 24 2c             mov    0x2c(%esp),%eax
8048b9a:    ff 24 85 80 a2 04 08    jmp    *0x804a280(,%eax,4)
```

```
(gdb) x/8a 0x804a280
0x804a280:      0x8048ba1 <phase_3+79>  0x8048bc3 <phase_3+113> 0x8048be2 <phase_3+144> 0x8048c04 <phase_3+178>
0x804a290:      0x8048c1f <phase_3+205> 0x8048c37 <phase_3+229> 0x8048c52 <phase_3+256> 0x8048c6d <phase_3+283>
```

可知a<=7,故有八个跳转地址可供选择，通过查看0x804a280可得知

```
8048ba1:    b8 66 00 00 00          mov    $0x66,%eax
8048ba6:    81 7c 24 28 e6 01 00    cmpl   $0x1e6,0x28(%esp)
8048bad:    00
8048bae:    0f 84 de 00 00 00       je     8048c92 <phase_3+0x140>
8048bb4:    e8 36 06 00 00          call   80491ef <explode_bomb>

8048c92:    3a 44 24 27             cmp    0x27(%esp),%al
8048c96:    74 05                   je     8048c9d <phase_3+0x14b>
8048c98:    e8 52 05 00 00          call   80491ef <explode_bomb>
8048c9d:    83 c4 3c                add    $0x3c,%esp
8048ca0:    c3                      ret
```

假设输入a=0,则会跳转至8048ba1，看汇编可知b=0x66=f，c=0x1e6=486,其他情况与此类似

- 完成截图

```
0 f 486
Halfway there!
```

4. phase_4

- 思路

```
8048ceb:    83 ec 2c                sub    $0x2c,%esp
8048cee:    8d 44 24 1c             lea    0x1c(%esp),%eax
8048cf2:    89 44 24 0c             mov    %eax,0xc(%esp)
8048cf6:    8d 44 24 18             lea    0x18(%esp),%eax
8048cfa:    89 44 24 08             mov    %eax,0x8(%esp)
8048cfe:    c7 44 24 04 b1 a4 04    movl   $0x804a4b1,0x4(%esp)
```

```
(gdb) x/s 0x804a4b1
0x804a4b1:      "%d %d"
```

查看0x804a4b1可知需要输入两个整数，设为a,b

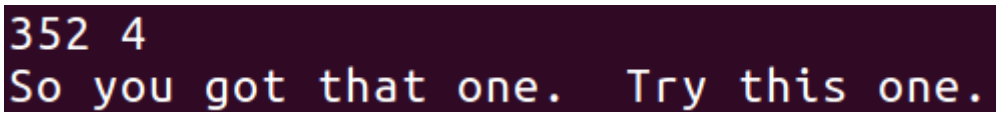
```
8048d17:    8b 44 24 1c             mov    0x1c(%esp),%eax
8048d1b:    83 e8 02                sub    $0x2,%eax
8048d1e:    83 f8 02                cmp    $0x2,%eax
8048d21:    76 05                   jbe    8048d28 <phase_4+0x3d>
8048d23:    e8 c7 04 00 00          call   80491ef <explode_bomb>
8048d28:    8b 44 24 1c             mov    0x1c(%esp),%eax
8048d2c:    89 44 24 04             mov    %eax,0x4(%esp)
8048d30:    c7 04 24 09 00 00 00    movl   $0x9,(%esp)
8048d37:    e8 65 ff ff ff          call   8048ca1 <func4>
8048d3c:    3b 44 24 18             cmp    0x18(%esp),%eax
8048d40:    74 05                   je     8048d47 <phase_4+0x5c>
8048d42:    e8 a8 04 00 00          call   80491ef <explode_bomb>
```

由汇编可知b要大于等于4，a要等于func4(9,b)，func4的等效C语言代码如下：

```c
int func4(int a1, int a2) {
    if (a1 == 0) { return 0; }
    if (a1 == 1) { return a2; }
    return func4(a1 - 1, a2) + a2 + func4(a1 - 2, a2);
}
```

当b=4时，a=352

- 完成截图



5. phase_5

- 思路

```
8048d4b:    53                      push   %ebx
8048d4c:    83 ec 18                sub    $0x18,%esp
8048d4f:    8b 5c 24 20             mov    0x20(%esp),%ebx
8048d53:    89 1c 24                mov    %ebx,(%esp)
8048d56:    e8 60 02 00 00          call   8048fbb <string_length>
8048d5b:    83 f8 06                cmp    $0x6,%eax
8048d5e:    74 05                   je     8048d65 <phase_5+0x1a>
8048d60:    e8 8a 04 00 00          call   80491ef <explode_bomb>
```

由汇编可知需要输入一个字符串，并且字符串的长度一定得等于6，剩余等效C语言代码如下：

```c
for (i = 0; i < 6; i++) {
    char c = str[i] & 0xF;
    sum += *(0x804a2a0 + c * 4);
}
if (sum != 0x27) {
    explode_bomb();
}
```

可知c范围为0-15，查看0x804a2a0处的16个数字如下：

```
(gdb) x/16a 0x804a2a0
0x804a2a0 <array.3133>: 0x2     0xa     0x6     0x1
0x804a2b0 <array.3133+16>:      0xc     0x10    0x9     0x3
0x804a2c0 <array.3133+32>:      0x4     0x7     0xe     0x5
0x804a2d0 <array.3133+48>:      0xb     0x8     0xf     0xd
```

要选取6个数相加等于0x27,并选择6个字符的低四位为这六个数字的偏移量，其中一种答案如下

```
5*0x7+0x4=0x27
//对应下标为9和8，可选字符'9','8'
999998
```

- 完成截图

```
999998
Good work!  On to the next...
```

6. phase_6

- 思路

先是读入6个数字

- 第一部分

```
8048dad:   be 00 00 00 00          mov    $0x0,%esi
8048db2:   8b 44 b4 28             mov    0x28(%esp,%esi,4),%eax
8048db6:   83 e8 01                sub    $0x1,%eax
8048db9:   83 f8 05                cmp    $0x5,%eax
8048dbc:   76 05                   jbe    8048dc3 <phase_6+0x2f>
8048dbe:   e8 2c 04 00 00          call   80491ef <explode_bomb>
8048dc3:   83 c6 01                add    $0x1,%esi
8048dc6:   83 fe 06                cmp    $0x6,%esi
8048dc9:   74 1b                   je     8048de6 <phase_6+0x52>
8048dcb:   89 f3                   mov    %esi,%ebx
8048dcd:   8b 44 9c 28             mov    0x28(%esp,%ebx,4),%eax
8048dd1:   39 44 b4 24             cmp    %eax,0x24(%esp,%esi,4)
8048dd5:   75 05                   jne    8048ddc <phase_6+0x48>
8048dd7:   e8 13 04 00 00          call   80491ef <explode_bomb>
8048ddc:   83 c3 01                add    $0x1,%ebx
8048ddf:   83 fb 05                cmp    $0x5,%ebx
8048de2:   7e e9                   jle    8048dcd <phase_6+0x39>
8048de4:   eb cc                   jmp    8048db2 <phase_6+0x1e>
```

等价C语言代码如下：

```
    for (int i = 0; i < 6; i++) {
        if (nums[i] < 1 || nums[i] > 6) {
            explode_bomb();
        }
        for (int j = i + 1; j < 6; j++) {
            if (nums[i] == nums[j]) {
                explode_bomb();
            }
        }
    }
```

可知第一部分需要我们输入的六个数分别为1，2，3，4，5，6，未要求顺序

- 第二部分

```
8048de6:   8d 44 24 28          lea    0x28(%esp),%eax
8048dea:   8d 5c 24 40          lea    0x40(%esp),%ebx
8048dee:   b9 07 00 00 00       mov    $0x7,%ecx
8048df3:   89 ca                mov    %ecx,%edx
8048df5:   2b 10                sub    (%eax),%edx
8048df7:   89 10                mov    %edx,(%eax)
8048df9:   83 c0 04             add    $0x4,%eax
8048dfc:   39 d8                cmp    %ebx,%eax
8048dfe:   75 f3                jne    8048df3 <phase_6+0x5f>
```

等价C语言代码如下：

```
for (int i = 0; i < 6; i++){
    nums[i] = 7 - nums[i];
}
```

- 第三部分

```
8048e00:   bb 00 00 00 00       mov    $0x0,%ebx
8048e05:   eb 1d                jmp    8048e24 <phase_6+0x90>
8048e07:   8b 52 08             mov    0x8(%edx),%edx
8048e0a:   83 c0 01             add    $0x1,%eax
8048e0d:   39 c8                cmp    %ecx,%eax
8048e0f:   75 f6                jne    8048e07 <phase_6+0x73>
8048e11:   eb 05                jmp    8048e18 <phase_6+0x84>
8048e13:   ba 54 c1 04 08       mov    $0x804c154,%edx
8048e18:   89 54 b4 10          mov    %edx,0x10(%esp,%esi,4)
8048e1c:   83 c3 01             add    $0x1,%ebx
8048e1f:   83 fb 06             cmp    $0x6,%ebx
8048e22:   74 17                je     8048e3b <phase_6+0xa7>
8048e24:   89 de                mov    %ebx,%esi
8048e26:   8b 4c 9c 28          mov    0x28(%esp,%ebx,4),%ecx
8048e2a:   83 f9 01             cmp    $0x1,%ecx
8048e2d:   7e e4                jle    8048e13 <phase_6+0x7f>
8048e2f:   b8 01 00 00 00       mov    $0x1,%eax
8048e34:   ba 54 c1 04 08       mov    $0x804c154,%edx
8048e39:   eb cc                jmp    8048e07 <phase_6+0x73>
```

```
(gdb) x/3a 0x804c154
0x804c154 <node1>:      0x316   0x1     0x804c160 <node2>
(gdb) x/3a
0x804c160 <node2>:      0x3c4   0x2     0x804c16c <node3>
(gdb) x/3a
0x804c16c <node3>:      0x2e5   0x3     0x804c178 <node4>
(gdb) x/3a
0x804c178 <node4>:      0x36b   0x4     0x804c184 <node5>
(gdb) x/3a
0x804c184 <node5>:      0x19a   0x5     0x804c190 <node6>
(gdb) x/3a
0x804c190 <node6>:      0x3df   0x6     0x0
```

查看0x804c154可知此处是一个链表，且依次以1，2，3，4，5，6标记，节点结构体如下：

```
struct node{
    int val;
    int num;
    struct node* next;
}
```

等价C语言代码如下：

```
node *adr[6];
for (int i = 0; i < 6; i++) {
    node* first=0x804c154;
    if (nums[i] == 1) {
        adr[i] = first;
    } else {
        node *p = first->next;
        for (int j = 2; j <= 6; j++) {
            if (j == nums[i]) {
                adr[i] = p;
            }
            p = p->next;
        }
    }
}
```

可知它将nums[i]对应的节点的地址存储到adr[i]中

- 第四部分

```
8048e3b:   8b 5c 24 10          mov    0x10(%esp),%ebx
8048e3f:   8d 44 24 14          lea    0x14(%esp),%eax
8048e43:   8d 74 24 28          lea    0x28(%esp),%esi
8048e47:   89 d9                mov    %ebx,%ecx
8048e49:   8b 10                mov    (%eax),%edx
8048e4b:   89 51 08             mov    %edx,0x8(%ecx)
8048e4e:   83 c0 04             add    $0x4,%eax
8048e51:   39 f0                cmp    %esi,%eax
8048e53:   74 04                je     8048e59 <phase_6+0xc5>
8048e55:   89 d1                mov    %edx,%ecx
8048e57:   eb f0                jmp    8048e49 <phase_6+0xb5>
8048e59:   c7 42 08 00 00 00 00 movl   $0x0,0x8(%edx)
```

等价C语言代码如下：

```
for(int i = 0, j = 1; j <= 5; i++, j++){
    adr[i]->next = adr[j];
}
adr[5]->next=null;
```

将链表顺序改为输入数字的顺序

- 第五部分

```
8048e60:   be 05 00 00 00          mov    $0x5,%esi
8048e65:   8b 43 08                mov    0x8(%ebx),%eax
8048e68:   8b 00                   mov    (%eax),%eax
8048e6a:   39 03                   cmp    %eax,(%ebx)
8048e6c:   7d 05                   jge    8048e73 <phase_6+0xdf>
8048e6e:   e8 7c 03 00 00          call   80491ef <explode_bomb>
8048e73:   8b 5b 08                mov    0x8(%ebx),%ebx
8048e76:   83 ee 01                sub    $0x1,%esi
8048e79:   75 ea                   jne    8048e65 <phase_6+0xd1>
8048e7b:   83 c4 44                add    $0x44,%esp
8048e7e:   5b                      pop    %ebx
8048e7f:   5e                      pop    %esi
8048e80:   c3                      ret
```

等价C语言代码如下：

```
for(int i=0;i<5;i++){
    if(adr[i]->val<adr[i+1]->val){
        explode_bomb();
    }
}
```

确保链表中的val的值是递减的，查看可知最终答案为：

```
1 5 3 6 4 2
```

- 完成截图

```
1 5 3 6 4 2
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

7. 最终结果

- bomblab 完成截图

```
fujinlong@ubuntu:/mnt/hgfs/course/bomb42$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
0 f 486
Halfway there!
352 4
So you got that one.  Try this one.
999998
Good work!  On to the next...
1 5 3 6 4 2
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

- （可选）bomblab 隐藏关卡
- phase_defused

```
80493cd:   8d 44 24 30          lea    0x30(%esp),%eax
80493d1:   89 44 24 10          mov    %eax,0x10(%esp)
80493d5:   8d 44 24 28          lea    0x28(%esp),%eax
80493d9:   89 44 24 0c          mov    %eax,0xc(%esp)
80493dd:   8d 44 24 2c          lea    0x2c(%esp),%eax
80493e1:   89 44 24 08          mov    %eax,0x8(%esp)
80493e5:   c7 44 24 04 0b a5 04 movl   $0x804a50b,0x4(%esp)
80493ec:   08
80493ed:   c7 04 24 f0 c8 04 08 movl   $0x804c8f0,(%esp)
80493f4:   e8 d7 f3 ff ff       call   80487d0 <__isoc99_sscanf@plt>
80493f9:   83 f8 03             cmp    $0x3,%eax
80493fc:   75 35                jne    8049433 <phase_defused+0x81>
80493fe:   c7 44 24 04 14 a5 04 movl   $0x804a514,0x4(%esp)
8049405:   08
8049406:   8d 44 24 30          lea    0x30(%esp),%eax
804940a:   89 04 24             mov    %eax,(%esp)
804940d:   e8 c8 fb ff ff       call   8048fda <strings_not_equal>
8049412:   85 c0                test   %eax,%eax
8049414:   75 1d                jne    8049433 <phase_defused+0x81>
```

```
(gdb) x/s 0x804a50b
0x804a50b:      "%d %d %s"
(gdb) x/s 0x804a514
0x804a514:      "DrEvil"
```

查看可知输入格式，且字符串为额外输入，两个整数输入确定是第四关

- secret_phase

```
8048ed2:   53                   push   %ebx
8048ed3:   83 ec 18             sub    $0x18,%esp
8048ed6:   e8 a3 03 00 00       call   804927e <read_line>
8048edb:   c7 44 24 08 0a 00 00 movl   $0xa,0x8(%esp)
8048ee2:   00
8048ee3:   c7 44 24 04 00 00 00 movl   $0x0,0x4(%esp)
8048eea:   00
8048eeb:   89 04 24             mov    %eax,(%esp)
```

```
8048eee:   e8 3d f9 ff ff          call   8048830 <strtol@plt>
8048ef3:   89 c3                   mov    %eax,%ebx
8048ef5:   8d 40 ff                lea    -0x1(%eax),%eax
8048ef8:   3d e8 03 00 00          cmp    $0x3e8,%eax
8048efd:   76 05                   jbe    8048f04 <secret_phase+0x32>
8048eff:   e8 eb 02 00 00          call   80491ef <explode_bomb>
8048f04:   89 5c 24 04             mov    %ebx,0x4(%esp)
8048f08:   c7 04 24 a0 c0 04 08    movl   $0x804c0a0,(%esp)
8048f0f:   e8 6d ff ff ff          call   8048e81 <fun7>
8048f14:   83 f8 03                cmp    $0x3,%eax
8048f17:   74 05                   je     8048f1e <secret_phase+0x4c>
8048f19:   e8 d1 02 00 00          call   80491ef <explode_bomb>
8048f1e:   c7 04 24 44 a2 04 08    movl   $0x804a244,(%esp)
8048f25:   e8 56 f8 ff ff          call   8048780 <puts@plt>
8048f2a:   e8 83 04 00 00          call   80493b2 <phase_defused>
8048f2f:   83 c4 18                add    $0x18,%esp
8048f32:   5b                      pop    %ebx
8048f33:   c3                      ret
```

等价C语言为:

```
cin >> input;
if (input > 1001) {
    explode_bomb();
}
int result = fun7(0x804c0a0,input);
if (result != 3) {
    explode_bomb();
}
```

```
(gdb) x/3a 0x804c0a0
0x804c0a0 <n1>: 0x24    0x804c0ac <n21> 0x804c0b8 <n22>
(gdb) x/3a
0x804c0ac <n21>:        0x8     0x804c0dc <n31> 0x804c0c4 <n32>
(gdb) x/3a
0x804c0b8 <n22>:        0x32    0x804c0d0 <n33> 0x804c0e8 <n34>
(gdb) x/3a
0x804c0c4 <n32>:        0x16    0x804c130 <n43> 0x804c118 <n44>
(gdb) x/3a
0x804c0d0 <n33>:        0x2d    0x804c0f4 <n45> 0x804c13c <n46>
(gdb) x/3a
0x804c0dc <n31>:        0x6     0x804c100 <n41> 0x804c124 <n42>
(gdb) x/3a
0x804c0e8 <n34>:        0x6b    0x804c10c <n47> 0x804c148 <n48>
(gdb) x/3a
0x804c0f4 <n45>:        0x28    0x0     0x0
(gdb) x/3a
0x804c100 <n41>:        0x1     0x0     0x0
(gdb) x/3a
0x804c10c <n47>:        0x63    0x0     0x0
(gdb) x/3a
0x804c118 <n44>:        0x23    0x0     0x0
(gdb) x/3a
0x804c124 <n42>:        0x7     0x0     0x0
(gdb) x/3a
0x804c130 <n43>:        0x14    0x0     0x0
(gdb) x/3a
0x804c13c <n46>:        0x2f    0x0     0x0
(gdb) x/3a
0x804c148 <n48>:        0x3e9   0x0     0x0
```

查看0x804c0a0处，可分析出这是一个二叉树结构，传入的是二叉树根节点的地址

- fun7

```
08048e81 <fun7>:
 8048e81:   53                      push   %ebx
 8048e82:   83 ec 18                sub    $0x18,%esp
 8048e85:   8b 54 24 20             mov    0x20(%esp),%edx
 8048e89:   8b 4c 24 24             mov    0x24(%esp),%ecx
 8048e8d:   85 d2                   test   %edx,%edx
 8048e8f:   74 37                   je     8048ec8 <fun7+0x47>
 8048e91:   8b 1a                   mov    (%edx),%ebx
 8048e93:   39 cb                   cmp    %ecx,%ebx
 8048e95:   7e 13                   jle    8048eaa <fun7+0x29>
 8048e97:   89 4c 24 04             mov    %ecx,0x4(%esp)
 8048e9b:   8b 42 04                mov    0x4(%edx),%eax
 8048e9e:   89 04 24                mov    %eax,(%esp)
 8048ea1:   e8 db ff ff ff          call   8048e81 <fun7>
 8048ea6:   01 c0                   add    %eax,%eax
 8048ea8:   eb 23                   jmp    8048ecd <fun7+0x4c>
 8048eaa:   b8 00 00 00 00          mov    $0x0,%eax
 8048eaf:   39 cb                   cmp    %ecx,%ebx
 8048eb1:   74 1a                   je     8048ecd <fun7+0x4c>
 8048eb3:   89 4c 24 04             mov    %ecx,0x4(%esp)
 8048eb7:   8b 42 08                mov    0x8(%edx),%eax
 8048eba:   89 04 24                mov    %eax,(%esp)
 8048ebd:   e8 bf ff ff ff          call   8048e81 <fun7>
 8048ec2:   8d 44 00 01             lea    0x1(%eax,%eax,1),%eax
 8048ec6:   eb 05                   jmp    8048ecd <fun7+0x4c>
 8048ec8:   b8 ff ff ff ff          mov    $0xffffffff,%eax
```

```
8048ecd:    83 c4 18                add    $0x18,%esp
8048ed0:    5b                      pop    %ebx
8048ed1:    c3                      ret
```

等价C语言代码如下：

```c
int fun7(Tree* root, int x) {
    if (!root)
        return -1;
    if (root->val == x)
        return 0;
    else if (root->val < x)
        return 2 * fun7(root -> right, x) + 1;
    else
        return 2 * fun7(root -> left, x);
}
```

最后结果应该返回3,可逆推x为根节点的右子树的右子树的根节点的值，即107

- 运行结果

```
fujinlong@ubuntu:/mnt/hgfs/course/bomb42$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
0 f 486
Halfway there!
352 4 DrEvil
So you got that one.  Try this one.
999998
Good work!  On to the next...
1 5 3 6 4 2
Curses, you've found the secret phase!
But finding it and solving it are quite different...
107
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

8. 备注

助教真帅