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：赵维康
学号：161630220
班级：1616302
指导教师：李博涵
实验日期：2018/6/7
我的炸弹编号：bomb21
我的答案：

```
Border relations with Canada have never been better.
8 8 24 56 136 328
0 85
5 2 HappyHangHangXueZhang
5 115
2 3 4 5 1 6
50
```

第一关

考察内容

初步判断这一关考察的是什么
字符串比较

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
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 68 a1 04        movl    $0x804a168,0x4(%esp)//标准输入串的起始地址
 8048aed:  08
 8048aee:  8b 45 08                    mov     0x8(%ebp),%eax//待输入的字
```

符串地址

```
 8048af1:  89 04 24                mov    %eax,(%esp)
 8048af4:  e8 69 04 00 00          call   8048f62 <strings_not_equal>
 8048af9:  85 c0                   test   %eax,%eax//比较两者
 8048afb:  74 05                   je      8048b02 <phase_1+0x22>//相
等，转第二关
 8048afd:  e8 83 06 00 00          call    8049185 <explode_bomb>//否
则爆炸
 8048b02:  c9                      leave
 8048b03:  c3                      ret
```

操作过程或解题思路

如题，需要的话可以带截图

```
(gdb) x/s 0x804a168
0x804a168:      "Border relations with Canada have never been better."
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
```

根据对汇编代码的分析，查看 0x804a168 处，即得答案

第二关

考察内容

初步判断这一关考察的是什么

循环

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
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 a9 06 00 00          call   80491c7 <read_six_numbers>
 8048b1e:  8b 45 e0                mov    -0x20(%ebp),%eax
 8048b21:  83 f8 07                cmp    $0x7,%eax//第一个参数与 7 比
较，小于等于，引爆炸弹
 8048b24:  7e 05                   jle    8048b2b <phase_2+0x27>
 8048b26:  3b 45 e4                cmp     -0x1c(%ebp),%eax//第二个参
数和第一个参数相等
 8048b29:  7e 22                   jle    8048b4d <phase_2+0x49>
 8048b2b:  e8 55 06 00 00          call   8049185 <explode_bomb>
 8048b30:  eb 1b                   jmp    8048b4d <phase_2+0x49>
 8048b32:  8b 53 fc                mov    -0x4(%ebx),%edx//“前二个数”
 8048b35:  8b 43 f8                mov    -0x8(%ebx),%eax//“前一个数”
 8048b38:  8d 04 50                lea     (%eax,%edx,2),%eax//用于求
解第三个数（相对于前两个数，不是真正意义上的第三个数）
 8048b3b:  39 03                   cmp    %eax,(%ebx)
 8048b3d:  74 05                   je     8048b44 <phase_2+0x40>
 8048b3f:  e8 41 06 00 00          call   8049185 <explode_bomb>
 8048b44:  83 c3 04                add    $0x4,%ebx
 8048b47:  39 f3                   cmp    %esi,%ebx//控制循环次数
 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

如题，需要的话可以带截图

```
(gdb) b phase_2
Breakpoint 1 at 0x8048b09
(gdb) b explode_bomb
Breakpoint 2 at 0x804918b
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328

Breakpoint 1, 0x08048b09 in phase_2 ()
Missing separate debuginfos, use: debuginfo-install glibc-2.17-222.el7.i686
(gdb) continue
Continuing.
That's number 2.  Keep going!
```

如汇编代码的分析，假设第一个数是 8，第二个是也是 8，那么第三个数是第一个数加第二个数的两倍，即 24，余下的依次类推，输入后通过，结果如图。

第三关

考察内容

初步判断这一关考察的是什么
条件/分支

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
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//第一个参数
 8048b65:  89 44 24 0c              mov    %eax,0xc(%esp)
 8048b69:  8d 45 f4                 lea    -0xc(%ebp),%eax//第二个参数
 8048b6c:  89 44 24 08              mov    %eax,0x8(%esp)
 8048b70:  c7 44 24 04 11 a4 04     movl    $0x804a411,0x4(%esp)//输入的内容
 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>
 8048b83:  83 f8 01                 cmp    $0x1,%eax//参数个数大于 1
 8048b86:  7f 05                    jg     8048b8d <phase_3+0x31>
 8048b88:  e8 f8 05 00 00           call   8049185 <explode_bomb>
 8048b8d:  83 7d f4 07              cmpl    $0x7,-0xc(%ebp)//参数范围 0-7
 8048b91:  77 1f                    ja     8048bb2 <phase_3+0x56>
 8048b93:  8b 45 f4                 mov    -0xc(%ebp),%eax
 8048b96:  ff 24 85 e0 a1 04 08     jmp    *0x804a1e0(,%eax,4)
 8048b9d:  b8 8e 01 00 00           mov    $0x18e,%eax
 8048ba2:  eb 1f                    jmp    8048bc3 <phase_3+0x67>
 8048ba4:  b8 29 01 00 00           mov    $0x129,%eax
 8048ba9:  eb 18                    jmp    8048bc3 <phase_3+0x67>
```

```
 8048bab:  b8 a7 01 00 00          mov    $0x1a7,%eax
 8048bb0:  eb 11                   jmp    8048bc3 <phase_3+0x67>
 8048bb2:  e8 ce 05 00 00          call   8049185 <explode_bomb>
 8048bb7:  b8 00 00 00 00          mov    $0x0,%eax
 8048bbc:  eb 05                   jmp    8048bc3 <phase_3+0x67>
 8048bbe:  b8 55 01 00 00          mov    $0x155,%eax//参数 1 为 0 时，
跳转此处
 8048bc3:  83 e0 7f                and    $0x7f,%eax
 8048bc6:  3b 45 f0                cmp      -0x10(%ebp),%eax//与参数 2
比较，相等，解除，否则，引爆
 8048bc9:  74 26                   je     8048bf1 <phase_3+0x95>
 8048bcb:  e8 b5 05 00 00          call   8049185 <explode_bomb>
 8048bd0:  eb 1f                   jmp    8048bf1 <phase_3+0x95>
 8048bd2:  b8 98 02 00 00          mov    $0x298,%eax
 8048bd7:  eb 05                   jmp    8048bde <phase_3+0x82>
 8048bd9:  b8 0c 02 00 00          mov    $0x20c,%eax
 8048bde:  83 e0 c0                and    $0xffffffc0,%eax
 8048be1:  eb e3                   jmp    8048bc6 <phase_3+0x6a>
 8048be3:  b8 67 00 00 00          mov     $0x67,%eax//参数 1 为 1 时，
跳转此处
 8048be8:  eb d9                   jmp    8048bc3 <phase_3+0x67>
 8048bea:  b8 34 00 00 00          mov    $0x34,%eax
 8048bef:  eb d2                   jmp    8048bc3 <phase_3+0x67>
 8048bf1:  c9                      leave
 8048bf2:  c3                      ret
```

操作过程或解题思路

如题，需要的话可以带截图

```
(gdb) b phase_3
Breakpoint 1 at 0x8048b62
(gdb) b explode_bomb
Breakpoint 2 at 0x804918b
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
2 3

Breakpoint 1, 0x08048b62 in phase_3 ()
Missing separate debuginfos, use: debuginfo-install glibc-2.17-222.el7.i686
(gdb) p /x *0x804a1e0
$1 = 0x8048bbe
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/zhaoweikang/bomb21/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?

8 8 24 56 136 328
That's number 2.  Keep going!
0 85

Breakpoint 1, 0x08048b62 in phase_3 ()
(gdb) continue
Continuing.
Halfway there!
```

```
Phase 1 defused. How about the next one?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85

Breakpoint 1, 0x08048b62 in phase_3 ()
Missing separate debuginfos, use: debuginfo-install glibc-2.17-222.el7.i686
(gdb) p /x *0x804a1e4
$1 = 0x8048be3
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
1 103

Breakpoint 1, 0x08048b62 in phase_3 ()
(gdb) continue
Continuing.
Halfway there!
```

如图，由 x/s 0x804a411 得“%d，%d”，要输入两个数；由 p /x *0x804a1e0，得参数 1 为 0 时的跳转表为 0x8048bbe，从而求得参数 2 为 0x155&0x7f，是十进制 85，同样，可得到参数 1 为 1 时，参数 2 为 103，其余，依次类推，共得 7 组答案。

第四关

考察内容

初步判断这一关考察的是什么

递归调用和栈

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
08048bf3 <func4>:
 8048bf3:  55                          push   %ebp
 8048bf4:  89 e5                       mov    %esp,%ebp
 8048bf6:  56                          push   %esi
 8048bf7:  53                          push   %ebx
 8048bf8:  83 ec 10                    sub    $0x10,%esp
 8048bfb:  8b 55 08                    mov     0x8(%ebp),%edx//第一个参数
A
 8048bfe:  8b 45 0c                    mov    0xc(%ebp),%eax //第二个参数
B
 8048c01:  8b 5d 10                    mov    0x10(%ebp),%ebx//第三个参数
C
 8048c04:  89 d9                       mov    %ebx,%ecx
 8048c06:  29 c1                       sub    %eax,%ecx
 8048c08:  89 ce                       mov    %ecx,%esi
 8048c0a:  c1 ee 1f                    shr    $0x1f,%esi//右移 31 位
 8048c0d:  01 f1                       add    %esi,%ecx
 8048c0f:  d1 f9                       sar    %ecx//c/2
 8048c11:  01 c1                       add    %eax,%ecx//递归返回值加倍
 8048c13:  39 d1                       cmp    %edx,%ecx
 8048c15:  7e 17                      jle    8048c2e <func4+0x3b>//若 c>a，
c=c-1，进入递归
 8048c17:  83 e9 01                    sub    $0x1,%ecx
 8048c1a:  89 4c 24 08                 mov    %ecx,0x8(%esp)
 8048c1e:  89 44 24 04                 mov    %eax,0x4(%esp)
 8048c22:  89 14 24                    mov    %edx,(%esp)
 8048c25:  e8 c9 ff ff ff              call   8048bf3 <func4>
 8048c2a:  01 c0                       add    %eax,%eax
 8048c2c:  eb 20                       jmp    8048c4e <func4+0x5b>
 8048c2e:  b8 00 00 00 00              mov    $0x0,%eax//b=0
```

```
 8048c33:  39 d1                    cmp    %edx,%ecx
 8048c35:  7d 17                   jge    8048c4e <func4+0x5b>//若 c<a，
则 b=c+1，c=14，进入递归
 8048c37:  89 5c 24 08              mov    %ebx,0x8(%esp)
 8048c3b:  83 c1 01                 add    $0x1,%ecx
 8048c3e:  89 4c 24 04              mov    %ecx,0x4(%esp)
 8048c42:  89 14 24                 mov    %edx,(%esp)
 8048c45:  e8 a9 ff ff ff           call   8048bf3 <func4>
 8048c4a:  8d 44 00 01              lea    0x1(%eax,%eax,1),%eax//将递
归返回值加倍后再加 1
 8048c4e:  83 c4 10                 add    $0x10,%esp
 8048c51:  5b                       pop    %ebx
 8048c52:  5e                       pop    %esi
 8048c53:  5d                       pop    %ebp
 8048c54:  c3                       ret

08048c55 <phase_4>:
 8048c55:  55                       push   %ebp
 8048c56:  89 e5                    mov    %esp,%ebp
 8048c58:  83 ec 28                 sub    $0x28,%esp
 8048c5b:  8d 45 f0                 lea     -0x10(%ebp),%eax//第二个参
数
 8048c5e:  89 44 24 0c              mov    %eax,0xc(%esp)
 8048c62:  8d 45 f4                 lea    -0xc(%ebp),%eax
 8048c65:  89 44 24 08              mov    %eax,0x8(%esp)
 8048c69:  c7 44 24 04 11 a4 04     movl     $0x804a411,0x4(%esp)//p/x
输入两个数
 8048c70:  08
 8048c71:  8b 45 08                 mov    0x8(%ebp),%eax
 8048c74:  89 04 24                 mov    %eax,(%esp)
 8048c77:  e8 64 fb ff ff           call                          80487e0
<__isoc99_sscanf@plt>
 8048c7c:  83 f8 02                 cmp     $0x2,%eax//输入个数不为 2，
引爆
 8048c7f:  75 06                    jne    8048c87 <phase_4+0x32>
 8048c81:  83 7d f4 0e              cmpl   $0xe,-0xc(%ebp)//第一个数时
0-14，否则，引爆
 8048c85:  76 05                    jbe    8048c8c <phase_4+0x37>
 8048c87:  e8 f9 04 00 00           call   8049185 <explode_bomb>
 8048c8c:  c7 44 24 08 0e 00 00     movl   $0xe,0x8(%esp)//构造 fuc4 的
参数
 8048c93:  00
 8048c94:  c7 44 24 04 00 00 00     movl    $0x0,0x4(%esp)//  构造 fuc4
的参数
```

```
 8048c9b:  00
 8048c9c:  8b 45 f4                mov      -0xc(%ebp),%eax//构造 fuc4
的参数
 8048c9f:  89 04 24                mov     %eax,(%esp)
 8048ca2:  e8 4c ff ff ff          call    8048bf3 <func4>
 8048ca7:  83 f8 02                cmp     $0x2,%eax
 8048caa:  75 06                   jne     8048cb2 <phase_4+0x5d>
 8048cac:  83 7d f0 02             cmpl     $0x2,-0x10(%ebp)//fuc4 的返
回值为 2，否则，引爆
 8048cb0:  74 05                   je      8048cb7 <phase_4+0x62>
 8048cb2:  e8 ce 04 00 00          call    8049185 <explode_bomb>
 8048cb7:  c9                      leave
 8048cb8:  c3                      ret
```

操作过程或解题思路

如题，需要的话可以带截图

```
(gdb) b phase_4
Breakpoint 1 at 0x8048c5b
(gdb) b explode_bomb
Breakpoint 2 at 0x804918b
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85
Halfway there!
1 2

Breakpoint 1, 0x08048c5b in phase_4 ()
Missing separate debuginfos, use: debuginfo-install glibc-2.17-222.el7.i686
(gdb) x/s 0x804a411
0x804a411:      "%d %d"
```

```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85
Halfway there!
5 2

Breakpoint 1, 0x08048c5b in phase_4 ()
(gdb) continue
Continuing.
So you got that one.  Try this one.
```

正如汇编代码所分析的那样，由 x/s 0x804a411 得“%d，%d”,输入两个数，

由 cmpl　$0x2,-0x10(%ebp)知 fuc4 的返回值为 2，即第二个参数为 2，利用对 fuc4 的汇编代码的分析知，参数 1 的范围为 0-14，多次递归分析出参数 1 为 5，即答案为 5 2。

第五关

考察内容

初步判断这一关考察的是什么
　　指针

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
08048cb9 <phase_5>:
 8048cb9:  55                          push   %ebp
 8048cba:  89 e5                       mov    %esp,%ebp
 8048cbc:  83 ec 28                    sub    $0x28,%esp
 8048cbf:  8d 45 f0                    lea     -0x10(%ebp),%eax//第二个参数
 8048cc2:  89 44 24 0c                 mov    %eax,0xc(%esp)
 8048cc6:  8d 45 f4                    lea    -0xc(%ebp),%eax//第一个参数
 8048cc9:  89 44 24 08                 mov    %eax,0x8(%esp)
 8048ccd:  c7 44 24 04 11 a4 04        movl    $0x804a411,0x4(%esp)//由此得需输入两个数
 8048cd4:  08
 8048cd5:  8b 45 08                    mov    0x8(%ebp),%eax
 8048cd8:  89 04 24                    mov    %eax,(%esp)
 8048cdb:  e8 00 fb ff ff              call                          80487e0 <__isoc99_sscanf@plt>
 8048ce0:  83 f8 01                    cmp     $0x1,%eax//输入个数大于 1，否则，引爆
 8048ce3:  7f 05                       jg     8048cea <phase_5+0x31>
 8048ce5:  e8 9b 04 00 00              call   8049185 <explode_bomb>
 8048cea:  8b 45 f4                    mov    -0xc(%ebp),%eax
 8048ced:  83 e0 0f                    and    $0xf,%eax
 8048cf0:  89 45 f4                    mov    %eax,-0xc(%ebp)
 8048cf3:  83 f8 0f                    cmp      $0xf,%eax//第一个数不等于 15
 8048cf6:  74 28                       je     8048d20 <phase_5+0x67>
 8048cf8:  b9 00 00 00 00              mov    $0x0,%ecx
```

```
 8048cfd:  ba 00 00 00 00           mov    $0x0,%edx
 8048d02:  83 c2 01                 add    $0x1,%edx
 8048d05:  8b 04 85 00 a2 04 08     mov    0x804a200(,%eax,4),%eax//数
组的使用，将前一个元素的值作为下一个元素的下标
 8048d0c:  01 c1                    add    %eax,%ecx
 8048d0e:  83 f8 0f                 cmp    $0xf,%eax//循环 15 次
 8048d11:  75 ef                    jne    8048d02 <phase_5+0x49>
 8048d13:  89 45 f4                 mov    %eax,-0xc(%ebp)
 8048d16:  83 fa 0f                 cmp      $0xf,%edx//最后取出的数为
15
 8048d19:  75 05                    jne    8048d20 <phase_5+0x67>
 8048d1b:  3b 4d f0                 cmp     -0x10(%ebp),%ecx//叠加的结
果与输入的第二个数相等，否则引爆
 8048d1e:  74 05                    je     8048d25 <phase_5+0x6c>
 8048d20:  e8 60 04 00 00           call   8049185 <explode_bomb>
 8048d25:  c9                       leave
 8048d26:  c3                       ret
```

操作过程或解题思路

如题，需要的话可以带截图

```
(gdb) b phase_5
Breakpoint 1 at 0x8048cbf
(gdb) b explode_bomb
Breakpoint 2 at 0x804918b
(gdb) x/s 0x804a411
0x804a411:      "%d %d"
(gdb) p *0x804a200@16
$1 = {10, 2, 14, 7, 8, 12, 15, 11, 0, 4, 1, 13, 3, 9, 6, 5}
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85
Halfway there!
5 2
So you got that one.  Try this one.
5 115

Breakpoint 1, 0x08048cbf in phase_5 ()
Missing separate debuginfos, use: debuginfo-install glibc-2.17-222.el7.i686
(gdb) continue

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

正如汇编代码中所分析的，由 p/x 0x804a411 得“%d，%d”，输入两个数，再由 p *0x804a200 得如图所示得数组中的元素，又最后取出的数是 15，倒推 15 步得 12，其下标为 5，所以第一个数是 5，然后正推 15 步，依次得 12，3，7，

11，13，9，4，8，0，10，1，2，14，6，15，相加得第二个数为115。

第六关

考察内容

初步判断这一关考察的是什么
　　链表/指针/结构

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
08048d27 <phase_6>:
 8048d27:  55                          push   %ebp
 8048d28:  89 e5                       mov    %esp,%ebp
 8048d2a:  56                          push   %esi
 8048d2b:  53                          push   %ebx
 8048d2c:  83 ec 40                    sub    $0x40,%esp
 8048d2f:  8d 45 e0                    lea    -0x20(%ebp),%eax
 8048d32:  89 44 24 04                 mov    %eax,0x4(%esp)
 8048d36:  8b 45 08                    mov    0x8(%ebp),%eax
 8048d39:  89 04 24                    mov    %eax,(%esp)
 8048d3c:  e8 86 04 00 00              call                          80491c7
<read_six_numbers>//需要输入6个数
 8048d41:  be 00 00 00 00              mov    $0x0,%esi
 8048d46:  8b 44 b5 e0                 mov    -0x20(%ebp,%esi,4),%eax
 8048d4a:  83 e8 01                    sub    $0x1,%eax
 8048d4d:  83 f8 05                    cmp    $0x5,%eax//输入的6个数的范
围为0-6，否则，引爆
 8048d50:  76 05                       jbe    8048d57 <phase_6+0x30>
 8048d52:  e8 2e 04 00 00              call   8049185 <explode_bomb>
 8048d57:  83 c6 01                    add    $0x1,%esi
 8048d5a:  83 fe 06                    cmp    $0x6,%esi
 8048d5d:  75 07                       jne    8048d66 <phase_6+0x3f>
 8048d5f:  bb 00 00 00 00              mov    $0x0,%ebx
 8048d64:  eb 39                       jmp    8048d9f <phase_6+0x78>
 8048d66:  89 f3                       mov    %esi,%ebx
 8048d68:  8b 44 9d e0                 mov    -0x20(%ebp,%ebx,4),%eax
 8048d6c:  39 44 b5 dc                 cmp    %eax,-0x24(%ebp,%esi,4)
 8048d70:  75 05                       jne    8048d77 <phase_6+0x50>
 8048d72:  e8 0e 04 00 00              call   8049185 <explode_bomb>
 8048d77:  83 c3 01                    add    $0x1,%ebx
```

```
 8048d7a:  83 fb 05              cmp    $0x5,%ebx
 8048d7d:  7e e9                 jle    8048d68 <phase_6+0x41>
 8048d7f:  90                    nop
 8048d80:  eb c4                 jmp    8048d46 <phase_6+0x1f>
 8048d82:  8b 52 08              mov    0x8(%edx),%edx
 8048d85:  83 c0 01              add    $0x1,%eax
 8048d88:  39 c8                 cmp    %ecx,%eax
 8048d8a:  75 f6                 jne    8048d82 <phase_6+0x5b>
 8048d8c:  eb 05                 jmp    8048d93 <phase_6+0x6c>
 8048d8e:  ba 54 c1 04 08        mov    $0x804c154,%edx
 8048d93:  89 54 b5 c8           mov    %edx,-0x38(%ebp,%esi,4)
 8048d97:  83 c3 01              add    $0x1,%ebx
 8048d9a:  83 fb 06              cmp    $0x6,%ebx
 8048d9d:  74 17                 je     8048db6 <phase_6+0x8f>
 8048d9f:  89 de                 mov    %ebx,%esi
 8048da1:  8b 4c 9d e0           mov    -0x20(%ebp,%ebx,4),%ecx
 8048da5:  83 f9 01              cmp    $0x1,%ecx
 8048da8:  7e e4                 jle    8048d8e <phase_6+0x67>
 8048daa:  b8 01 00 00 00        mov    $0x1,%eax
 8048daf:  ba 54 c1 04 08        mov    $0x804c154,%edx//链表节点首
地址
 8048db4:  eb cc                 jmp    8048d82 <phase_6+0x5b>
 8048db6:  8b 5d c8              mov    -0x38(%ebp),%ebx
 8048db9:  8d 45 cc              lea    -0x34(%ebp),%eax
 8048dbc:  8d 75 e0              lea    -0x20(%ebp),%esi
 8048dbf:  89 d9                 mov    %ebx,%ecx
 8048dc1:  8b 10                 mov    (%eax),%edx
 8048dc3:  89 51 08              mov    %edx,0x8(%ecx)
 8048dc6:  83 c0 04              add    $0x4,%eax
 8048dc9:  39 f0                 cmp    %esi,%eax
 8048dcb:  74 04                 je     8048dd1 <phase_6+0xaa>
 8048dcd:  89 d1                 mov    %edx,%ecx
 8048dcf:  eb f0                 jmp    8048dc1 <phase_6+0x9a>
 8048dd1:  c7 42 08 00 00 00 00  movl   $0x0,0x8(%edx)
 8048dd8:  be 05 00 00 00        mov    $0x5,%esi//前一个数大于等于
后一个数
 8048ddd:  8b 43 08              mov     0x8(%ebx),%eax//前一个数大
于等于后一个数
 8048de0:  8b 00                 mov    (%eax),%eax//前一个数大于等
于后一个数
 8048de2:  39 03                 cmp    %eax,(%ebx)//否则，引爆
 8048de4:  7d 05                 jge    8048deb <phase_6+0xc4>
 8048de6:  e8 9a 03 00 00        call   8049185 <explode_bomb>
 8048deb:  8b 5b 08              mov    0x8(%ebx),%ebx
```

```
8048dee:  83 ee 01                sub    $0x1,%esi
8048df1:  75 ea                   jne    8048ddd <phase_6+0xb6>
8048df3:  83 c4 40                add    $0x40,%esp
8048df6:  5b                      pop    %ebx
8048df7:  5e                      pop    %esi
8048df8:  5d                      pop    %ebp
8048df9:  c3                      ret
```

操作过程或解题思路

如题，需要的话可以带截图

```
(gdb) b phase_6
Breakpoint 1 at 0x8048d2c
(gdb) b exp
expand_bkref_cache           expected3.7889
expand_dynamic_string_token  expected4.7890
expected.9599                expected_note.9423
expected1.7887               expected_note.9605
expected2.7888               explode_bomb
expected2.9598
(gdb) b explode_bomb
Breakpoint 2 at 0x804918b
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85
Halfway there!
5 2
So you got that one.  Try this one.
5 115
Good work!  On to the next...
1 2
Breakpoint 1, 0x08048d2c in phase_6 ()
(gdb) x 0x804c154
0x804c154 <node1>:      0x00000124
(gdb)
0x804c158 <node1+4>:    0x00000001
(gdb)
0x804c15c <node1+8>:    0x0804c160
(gdb) x 0x0804c160
0x804c160 <node2>:      0x00000383
(gdb)
0x804c164 <node2+4>:    0x00000002
(gdb)
0x804c168 <node2+8>:    0x0804c16c
(gdb) x 0x0804c16c
0x804c16c <node3>:      0x000002a0
(gdb)
0x804c170 <node3+4>:    0x00000003
(gdb)
0x804c174 <node3+8>:    0x0804c178
(gdb) x 0x0804c178
0x804c178 <node4>:      0x00000280
(gdb)
0x804c17c <node4+4>:    0x00000004
(gdb)
0x804c180 <node4+8>:    0x0804c184
(gdb) x 0x0804c184
```

```
0x804c184 <node5>:      0x00000227
(gdb)
0x804c188 <node5+4>:    0x00000005
(gdb)
0x804c18c <node5+8>:    0x0804c190
(gdb) x 0x0804c190
0x804c190 <node6>:      0x000000ec
(gdb)
0x804c194 <node6+4>:    0x00000006
(gdb)
0x804c198 <node6+8>:    0x00000000
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85
Halfway there!
5 2
So you got that one.  Try this one.
5 115

Good work!  On to the next...
2 3 4 5 1 6

Breakpoint 1, 0x08048d2c in phase_6 ()
(gdb) continue
Continuing.
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
[Inferior 1 (process 6099) exited normally]
(gdb)
```

正如汇编代码所分析的那样，需要输入 6 个数，由 x 0x804c154 知，链表的每个节点存储了两个 int 型的数据，以及指向下一个节点的指针。如上图所示，依次得到 6 个节点中所存储的值为 124、383、2a0、280、227、ec，由汇编代码分析知，需从大到小排序，为 383>2a0>280>227>124> ec。它们对应的顺序为 2 3 4 5 1 6，即为答案。

隐藏关

考察内容

初步判断这一关考察的是什么

二叉树/递归

进入本关的方法

描述一下你是从何得知并用何方法进入本隐藏关的

在 phase_defused 中，由 x/s 0x804a46b 得“%d，%d，%s”，即在某一关后加一个字符串，即可开启隐藏关，由 x/s 0x804a474 得，对应的字符串为“Happy HangHangXueZhang”，由 x/s 0x804a2c0 得开启隐藏关后的提示语“Curses,you ’ve found the secret phase!”,经过计算，在第四关后加字符串“Happy HangHangXueZhang”即可开启隐藏关。

反汇编代码分析

附汇编代码，并通过适当在代码中穿插注释来理解代码

```
08048dfa <fun7>:
 8048dfa:  55                          push   %ebp
 8048dfb:  89 e5                       mov    %esp,%ebp
 8048dfd:  53                          push   %ebx
 8048dfe:  83 ec 14                    sub    $0x14,%esp
 8048e01:  8b 55 08                    mov    0x8(%ebp),%edx //第一个参数
a
 8048e04:  8b 4d 0c                    mov    0xc(%ebp),%ecx //第二个参数
b
 8048e07:  85 d2                       test   %edx,%edx//内层递归为 0
 8048e09:  74 37                       je     8048e42 <fun7+0x48>
 8048e0b:  8b 1a                       mov    (%edx),%ebx
 8048e0d:  39 cb                       cmp    %ecx,%ebx
 8048e0f:  7e 13                      jle    8048e24 <fun7+0x2a> //若 a>b，
将(‘a’+4)作为地址进入递归
 8048e11:  89 4c 24 04                 mov    %ecx,0x4(%esp)
 8048e15:  8b 42 04                    mov    0x4(%edx),%eax
 8048e18:  89 04 24                    mov    %eax,(%esp)
 8048e1b:  e8 da ff ff ff              call   8048dfa <fun7>
 8048e20:  01 c0                       add    %eax,%eax//在此处将递归返回
值加倍
 8048e22:  eb 23                       jmp    8048e47 <fun7+0x4d>
 8048e24:  b8 00 00 00 00              mov    $0x0,%eax
 8048e29:  39 cb                       cmp    %ecx,%ebx
 8048e2b:  74 1a                       je     8048e47 <fun7+0x4d>
 8048e2d:  89 4c 24 04                 mov     %ecx,0x4(%esp) //若 a<B，将
(‘a’+8)作为地址进入递归
 8048e31:  8b 42 08                    mov    0x8(%edx),%eax
 8048e34:  89 04 24                    mov    %eax,(%esp)
 8048e37:  e8 be ff ff ff              call   8048dfa <fun7>
 8048e3c:  8d 44 00 01                 lea     0x1(%eax,%eax,1),%eax //在
此处将递归返回值加倍后再加 1
```

```
 8048e40:  eb 05                    jmp    8048e47 <fun7+0x4d>
 8048e42:  b8 ff ff ff ff           mov    $0xffffffff,%eax
 8048e47:  83 c4 14                 add    $0x14,%esp
 8048e4a:  5b                       pop    %ebx
 8048e4b:  5d                       pop    %ebp
 8048e4c:  c3                       ret

08048e4d <secret_phase>:
 8048e4d:  55                       push   %ebp
 8048e4e:  89 e5                    mov    %esp,%ebp
 8048e50:  53                       push   %ebx
 8048e51:  83 ec 14                 sub    $0x14,%esp
 8048e54:  e8 bd 03 00 00           call   8049216 <read_line>
 8048e59:  c7 44 24 08 0a 00 00     movl    $0xa,0x8(%esp)//输入一个十
进制数
 8048e60:  00
 8048e61:  c7 44 24 04 00 00 00     movl   $0x0,0x4(%esp)// 输入一个十
进制数
 8048e68:  00
 8048e69:  89 04 24                 mov    %eax,(%esp)
 8048e6c:  e8 cf f9 ff ff           call    8048840 <strtol@plt>//将字
符串转换为一个数字
 8048e71:  89 c3                    mov    %eax,%ebx
 8048e73:  8d 40 ff                 lea    -0x1(%eax),%eax
 8048e76:  3d e8 03 00 00           cmp    $0x3e8,%eax//输入的数要小于
1001，否则，引爆
 8048e7b:  76 05                    jbe    8048e82 <secret_phase+0x35>
 8048e7d:  e8 03 03 00 00           call   8049185 <explode_bomb>
 8048e82:  89 5c 24 04              mov    %ebx,0x4(%esp)
 8048e86:  c7 04 24 a0 c0 04 08     movl   $0x804c0a0,(%esp)
 8048e8d:  e8 68 ff ff ff           call   8048dfa <fun7>
 8048e92:  83 f8 01                 cmp    $0x1,%eax//fun7 的返回值为 1
 8048e95:  74 05                    je     8048e9c <secret_phase+0x4f>
 8048e97:  e8 e9 02 00 00           call   8049185 <explode_bomb>
 8048e9c:  c7 04 24 a0 a1 04 08     movl   $0x804a1a0,(%esp)
 8048ea3:  e8 d8 f8 ff ff           call   8048780 <puts@plt>
 8048ea8:  e8 a1 04 00 00           call   804934e <phase_defused>
 8048ead:  83 c4 14                 add    $0x14,%esp
 8048eb0:  5b                       pop    %ebx
 8048eb1:  5d                       pop    %ebp
 8048eb2:  c3                       ret
 8048eb3:  66 90                    xchg   %ax,%ax
 8048eb5:  66 90                    xchg   %ax,%ax
 8048eb7:  66 90                    xchg   %ax,%ax
```

```
 8048eb9:  66 90                    xchg   %ax,%ax
 8048ebb:  66 90                    xchg   %ax,%ax
 8048ebd:  66 90                    xchg   %ax,%ax
 8048ebf:  90                       nop

0804934e <phase_defused>:
 804934e:  55                       push   %ebp
 804934f:  89 e5                    mov    %esp,%ebp
 8049351:  81 ec 88 00 00 00        sub    $0x88,%esp
 8049357:  c7 04 24 01 00 00 00     movl   $0x1,(%esp)
 804935e:  e8 4b fd ff ff           call   80490ae <send_msg>
 8049363:  83 3d e8 c7 04 08 06     cmpl     $0x6,0x804c7e8//通过前面 6
关，才可进入隐藏关
 804936a:  75 7a                   jne    80493e6 <phase_defused+0x98>
 804936c:  8d 45 a8                 lea    -0x58(%ebp),%eax
 804936f:  89 44 24 10              mov    %eax,0x10(%esp)
 8049373:  8d 45 a0                 lea    -0x60(%ebp),%eax
 8049376:  89 44 24 0c              mov    %eax,0xc(%esp)
 804937a:  8d 45 a4                 lea    -0x5c(%ebp),%eax
 804937d:  89 44 24 08              mov    %eax,0x8(%esp)
 8049381:  c7 44 24 04 6b a4 04     movl     $0x804a46b,0x4(%esp)//输入
为“%d，%d，%s”
 8049388:  08
 8049389:  c7 04 24 f0 c8 04 08     movl   $0x804c8f0,(%esp)//提示输入
的是字符串
 8049390:  e8 4b f4 ff ff           call                            80487e0
<__isoc99_sscanf@plt>
 8049395:  83 f8 03                 cmp    $0x3,%eax
 8049398:  75 34                   jne    80493ce <phase_defused+0x80>
 804939a:  c7 44 24 04 74 a4 04     movl     $0x804a474,0x4(%esp)//开启
隐藏关的字符串内容
 80493a1:  08
 80493a2:  8d 45 a8                 lea    -0x58(%ebp),%eax
 80493a5:  89 04 24                 mov    %eax,(%esp)
 80493a8:  e8 b5 fb ff ff           call   8048f62 <strings_not_equal>
 80493ad:  85 c0                    test   %eax,%eax
 80493af:  75 1d                   jne    80493ce <phase_defused+0x80>
 80493b1:  c7 04 24 c0 a2 04 08     movl   $0x804a2c0,(%esp)//若相等，
出现提示语句
 80493b8:  e8 c3 f3 ff ff           call   8048780 <puts@plt>
 80493bd:  c7 04 24 e8 a2 04 08     movl   $0x804a2e8,(%esp)
 80493c4:  e8 b7 f3 ff ff           call   8048780 <puts@plt>
 80493c9:  e8 7f fa ff ff           call   8048e4d <secret_phase>
 80493ce:  c7 04 24 20 a3 04 08     movl   $0x804a320,(%esp)
```

```
80493d5:  e8 a6 f3 ff ff          call   8048780 <puts@plt>
80493da:  c7 04 24 4c a3 04 08    movl   $0x804a34c,(%esp)
80493e1:  e8 9a f3 ff ff          call   8048780 <puts@plt>
80493e6:  c9                      leave
80493e7:  c3                      ret
80493e8:  66 90                   xchg   %ax,%ax
80493ea:  66 90                   xchg   %ax,%ax
80493ec:  66 90                   xchg   %ax,%ax
80493ee:  66 90                   xchg   %ax,%ax
```

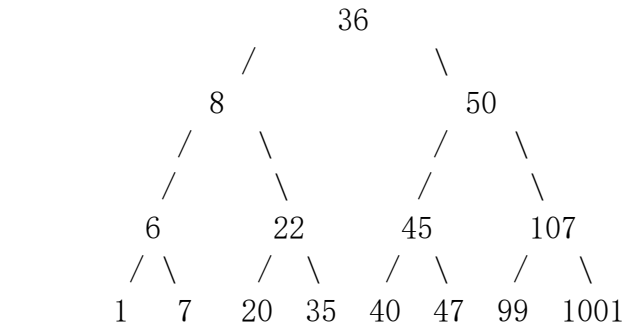
操作过程或解题思路

如题，需要的话可以带截图

```
(gdb) b explode_bomb
Breakpoint 1 at 0x804918b
(gdb) x/s 0x804a46b
0x804a46b:      "%d %d %s"
(gdb) x/s 0x804c8f0
0x804c8f0 <input_strings+240>:  ""
(gdb) x/s 0x804a474
0x804a474:      "HappyHangHangXueZhang"
(gdb) x/s 0x804a2c0
0x804a2c0:      "Curses, you've found the secret phase!"
(gdb) x/120a 0x804c0a0
0x804c0a0 <n1>: 0x24    0x804c0ac <n21> 0x804c0b8 <n22> 0x8
0x804c0b0 <n21+4>:      0x804c0dc <n31> 0x804c0c4 <n32> 0x32    0x804c0d0 <n33>
0x804c0c0 <n22+8>:      0x804c0e8 <n34> 0x16    0x804c130 <n43> 0x804c118 <n44>
0x804c0d0 <n33>:        0x2d    0x804c0f4 <n45> 0x804c13c <n46> 0x6
0x804c0e0 <n31+4>:      0x804c100 <n41> 0x804c124 <n42> 0x6b    0x804c10c <n47>
0x804c0f0 <n34+8>:      0x804c148 <n48> 0x28    0x0     0x0
0x804c100 <n41>:        0x1     0x0     0x0     0x63
0x804c110 <n47+4>:      0x0     0x0     0x23    0x0
0x804c120 <n44+8>:      0x0     0x7     0x0     0x0
0x804c130 <n43>:        0x14    0x0     0x0     0x2f
0x804c140 <n46+4>:      0x0     0x0     0x3e9   0x0
0x804c150 <n48+8>:      0x0     0x124   0x1     0x804c160 <node2>
0x804c160 <node2>:      0x383   0x2     0x804c16c <node3>       0x2a0
0x804c170 <node3+4>:    0x3     0x804c178 <node4>       0x280   0x4
0x804c180 <node4+8>:    0x804c184 <node5>       0x227   0x5     0x804c190 <node6>
0x804c190 <node6>:      0xec    0x6     0x0     0x0
0x804c1a0 <userid>:     0x36313631      0x32323033      0x30    0x0
0x804c1b0 <userid+16>:  0x0     0x0     0x0     0x0
0x804c1c0 <userid+32>:  0x0     0x0     0x0     0x0
0x804c1d0 <userid+48>:  0x0     0x0     0x0     0x0
0x804c1e0 <userid+64>:  0x0     0x0     0x0     0x0
0x804c1f0 <userid+80>:  0x0     0x0     0x0     0x0
0x804c200 <userid+96>:  0x0     0x0     0x0     0x0
0x804c210 <userid+112>: 0x0     0x0     0x0     0x0
0x804c220 <userid+128>: 0x0     0x0     0x0     0x0
---Type <return> to continue, or q <return> to quit---q
```

```
(gdb) run
Starting program: /home/zhaoweikang/bomb21/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?
8 8 24 56 136 328
That's number 2.  Keep going!
0 85
Halfway there!
5 2 HappyHangHangXueZhang
So you got that one.  Try this one.
5 115
Good work!  On to the next...
2 3 4 5 1 6
Curses, you've found the secret phase!
But finding it and solving it are quite different...
50
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
[Inferior 1 (process 8860) exited normally]
Missing separate debuginfos, use: debuginfo-install glibc-2.17-222.el7.i686
(gdb)
```

正如汇编代码所分析的那样，由 x/120a 0x804c0a0 可以得到，内存中存储的二叉树的各个节点，即



而 fun7 可以用 c 语言重写为

```
struct treeNode
{
    int data;
    struct treeNode* leftChild;
    struct treeNode* rightChild;
};

int fun7(struct treeNode* p, int v)
{
    if (p == NULL)
        return -1;
    else if (v < p->data)
        return 2 * fun7(p->leftChild, v);
    else if (v == p->data)
        return 0;
    else
        return 2 * fun7(p->rightChild, v) + 1;
```

}
要想返回值为 1，则其返回的模式必须为 1+2*0，从而递归求出所求值为 50。

思考与体会

简单说一说自己在完成的过程中遇到的困难或者技巧等，或者学到了什么

首先要学会 gdb 调试，这是基本技能，必须过关，然后要能看的懂汇编代码，从汇编代码入手，进行分析，从而得到求解每关的信息。其次，学会设置断点及调试是极为重要的，这样的话，就几乎不会爆炸了。