201824636 이강우

Phase_1

```
201824636@CSEDell: ~/BombLab/bomb48
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bomb...done.
(gdb) break phase_1
Breakpoint 1 at 0x1264
(gdb) break explode_bomb()
Function "explode_bomb()" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) break explode_bomb
Breakpoint 2 at 0x1a69
(gdb) disas phase_1
Dump of assembler code for function phase_1:
   0x0000000000001264 <+0>:     sub    $0x8,%rsp
   0x0000000000001268 <+4>:     lea    0x1831(%rip),%rsi        # 0x2aa0
   0x000000000000126f <+11>:    callq  0x1765 <strings_not_equal>
   0x0000000000001274 <+16>:    test   %eax,%eax
   0x0000000000001276 <+18>:    jne    0x127d <phase_1+25>
   0x0000000000001278 <+20>:    add    $0x8,%rsp
   0x000000000000127c <+24>:    retq
   0x000000000000127d <+25>:    callq  0x1a69 <explode_bomb>
   0x0000000000001282 <+30>:    jmp    0x1278 <phase_1+20>
End of assembler dump.
(gdb)
```

disassemble하여 코드를 보면 strings_not_equal 함수를 호출한다. 이것으로 문자열을 받는다는 사실을 알고 함수의 리턴이 같지 않으면 explode_bomb으로 이동하여 터진다는 것을 알 수 있다.

rdi에는 사용자 입력값 rsi는 이미 저장된 문자열을 비교한다는 사실을 알아 조회해보니 답으로 입력해야하는 문자열이 출력되어 해결 하였다.

```
201824636@CSEDell: ~/BombLab/bomb48
rbp            0x5555555568d0   0x5555555568d0 <__libc_csu_init>
rsp            0x7fffffffe468   0x7fffffffe468
r8             0x555555759e62   93824994352738
r9             0x7ffff7fe5540   140737354028352
r10            0x555555759010   93824994349072
r11            0x246    582
r12            0x555555555000   93824992235520
r13            0x7fffffffe560   140737488348512
r14            0x0      0
r15            0x0      0
rip            0x555555555765   0x555555555765 <strings_not_equal>
eflags         0x202    [ IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
---Type <return> to continue, or q <return> to quit---q
Quit
(gdb) x/s $rdi
0x5555557586c0 <input_strings>: "c"
(gdb) x/s $rsi
0x555555556aa0: "For NASA, space is still a high priority."
(gdb)
```

```
201824636@CSEDell: ~/BombLab/bomb48
Dump of assembler code for function phase_2:
   0x0000555555555284 <+0>:     push   %rbp
   0x0000555555555285 <+1>:     push   %rbx
   0x0000555555555286 <+2>:     sub    $0x28,%rsp
   0x000055555555528a <+6>:     mov    %fs:0x28,%rax
   0x0000555555555293 <+15>:    mov    %rax,0x18(%rsp)
   0x0000555555555298 <+20>:    xor    %eax,%eax
   0x000055555555529a <+22>:    mov    %rsp,%rsi
   0x000055555555529d <+25>:    callq  0x555555555aa5 <read_six_n
   0x00005555555552a2 <+30>:    cmpl   $0x1,(%rsp)
   0x00005555555552a6 <+34>:    jne    0x5555555552b1 <phase_2+45
   0x00005555555552a8 <+36>:    mov    %rsp,%rbx
   0x00005555555552ab <+39>:    lea    0x14(%rbx),%rbp
   0x00005555555552af <+43>:    jmp    0x5555555552c1 <phase_2+61
   0x00005555555552b1 <+45>:    callq  0x555555555a69 <explode_bo
   0x00005555555552b6 <+50>:    jmp    0x5555555552a8 <phase_2+36
   0x00005555555552b8 <+52>:    add    $0x4,%rbx
   0x00005555555552bc <+56>:    cmp    %rbp,%rbx
   0x00005555555552bf <+59>:    je     0x5555555552d1 <phase_2+77
   0x00005555555552c1 <+61>:    mov    (%rbx),%eax
   0x00005555555552c3 <+63>:    add    %eax,%eax
   0x00005555555552c5 <+65>:    cmp    %eax,0x4(%rbx)
   0x00005555555552c8 <+68>:    je     0x5555555552b8 <phase_2+52
---Type <return> to continue, or q <return> to quit---
   0x00005555555552ca <+70>:    callq  0x555555555a69 <explode_bo
   0x00005555555552cf <+75>:    jmp    0x5555555552b8 <phase_2+52
   0x00005555555552d1 <+77>:    mov    0x18(%rsp),%rax
   0x00005555555552d6 <+82>:    xor    %fs:0x28,%rax
   0x00005555555552df <+91>:    jne    0x5555555552e8 <phase_2+10
   0x00005555555552e1 <+93>:    add    $0x28,%rsp
   0x00005555555552e5 <+97>:    pop    %rbx
   0x00005555555552e6 <+98>:    pop    %rbp
   0x00005555555552e7 <+99>:    retq
   0x00005555555552e8 <+100>:   callq  0x555555554ea0 <__stack_ch
End of assembler dump.
(gdb) c
Continuing.

Breakpoint 5, 0x0000555555555765 in strings_not_equal ()
(gdb) c
Continuing.
Phase 1 defused. How about the next one?
1 2 4 8 16 32

Breakpoint 4, 0x0000555555555284 in phase_2 ()
(gdb) c
Continuing.

Breakpoint 6, 0x00005555555555aa5 in read_six_numbers ()
(gdb) c
Continuing.
That's number 2.  Keep going!
^C
```

Phase_2

+25라인에 read_six_numbers로 유추해볼 때 6개의 숫자를 입력한다는 걸 알 수 있었다.

+30라인에 rsp가 1과 비교해서 같지않으면 bomb이터지는것으로 보아 첫번 째 숫자는 1이다.

그 후 +63라인에서 eax+eax를 하고 je를 통해 비교하고 +52로 jmp하는 것으로 보아 현재 값 = 전 입력값끼리 를 더해주는 반복문이라고 예상하여. 답을 입력하였다.

Phase_3

```
(gdb) ni
0x00005555555552f1 in phase_3 ()
(gdb) x/s 0x555555556d7d
0x555555556d7d: "%d %d"
(gdb)
```

입력받기 전의 주소를 검사해 입력값은 %d %d의 변수인 것을 확인한다.

```
Breakpoint 9, 0x00005555555552ed in phase_3 ()
(gdb) disas phase_3
Dump of assembler code for function phase_3:
=> 0x00005555555552ed <+0>:     sub    $0x18,%rsp
   0x00005555555552f1 <+4>:     mov    %fs:0x28,%rax
   0x00005555555552fa <+13>:    mov    %rax,0x8(%rsp)
   0x00005555555552ff <+18>:    xor    %eax,%eax
   0x0000555555555301 <+20>:    lea    0x4(%rsp),%rcx
   0x0000555555555306 <+25>:    mov    %rsp,%rdx
   0x0000555555555309 <+28>:    lea    0x1a6d(%rip),%rsi        # 0x555555556d7d
   0x0000555555555310 <+35>:    callq  0x55555554f40 <__isoc99_sscanf@plt>
   0x0000555555555315 <+40>:    cmp    $0x1,%eax
   0x0000555555555318 <+43>:    jle    0x555555555337 <phase_3+74>
   0x000055555555531a <+45>:    cmpl   $0x7,(%rsp)
   0x000055555555531e <+49>:    ja     0x5555555553bd <phase_3+208>
   0x0000555555555324 <+55>:    mov    (%rsp),%eax
   0x0000555555555327 <+58>:    lea    0x17d2(%rip),%rdx        # 0x555555556b00
   0x000055555555532e <+65>:    movslq (%rdx,%rax,4),%rax
   0x0000555555555332 <+69>:    add    %rdx,%rax
   0x0000555555555335 <+72>:    jmpq   *%rax
   0x0000555555555337 <+74>:    callq  0x555555555a69 <explode_bomb>
   0x000055555555533c <+79>:    jmp    0x555555555531a <phase_3+45>
   0x000055555555533e <+81>:    mov    $0x1c0,%eax
   0x0000555555555343 <+86>:    jmp    0x55555555534a <phase_3+93>
   0x0000555555555345 <+88>:    mov    $0x0,%eax
   0x000055555555534a <+93>:    sub    $0x1b5,%eax
   0x000055555555534f <+98>:    add    $0x15d,%eax
   0x0000555555555354 <+103>:   sub    $0x38b,%eax
   0x0000555555555359 <+108>:   add    $0x38b,%eax
   0x000055555555535e <+113>:   sub    $0x38b,%eax
   0x0000555555555363 <+118>:   add    $0x38b,%eax
   0x0000555555555368 <+123>:   sub    $0x38b,%eax
   0x000055555555536d <+128>:   cmpl   $0x5,(%rsp)
   0x0000555555555371 <+132>:   jg     0x555555555379 <phase_3+140>
   0x0000555555555373 <+134>:   cmp    %eax,0x4(%rsp)
   0x0000555555555377 <+138>:   je     0x55555555537e <phase_3+145>
   0x0000555555555379 <+140>:   callq  0x555555555a69 <explode_bomb>
   0x000055555555537e <+145>:   mov    0x8(%rsp),%rax
   0x0000555555555383 <+150>:   xor    %fs:0x28,%rax
   0x000055555555538c <+159>:   jne    0x5555555553c9 <phase_3+220>
   0x000055555555538e <+161>:   add    $0x18,%rsp
   0x0000555555555392 <+165>:   retq
   0x0000555555555393 <+166>:   mov    $0x0,%eax
   0x0000555555555398 <+171>:   jmp    0x55555555534f <phase_3+98>
   0x000055555555539a <+173>:   mov    $0x0,%eax
   0x000055555555539f <+178>:   jmp    0x555555555354 <phase_3+103>
   0x00005555555553a1 <+180>:   mov    $0x0,%eax
   0x00005555555553a6 <+185>:   jmp    0x555555555359 <phase_3+108>
   0x00005555555553a8 <+187>:   mov    $0x0,%eax
   0x00005555555553ad <+192>:   jmp    0x55555555535e <phase_3+113>
   0x00005555555553af <+194>:   mov    $0x0,%eax
   0x00005555555553b4 <+199>:   jmp    0x555555555363 <phase_3+118>
   0x00005555555553b6 <+201>:   mov    $0x0,%eax
   0x00005555555553bb <+206>:   jmp    0x555555555368 <phase_3+123>
   0x00005555555553bd <+208>:   callq  0x555555555a69 <explode_bomb>
   0x00005555555553c2 <+213>:   mov    $0x0,%eax
   0x00005555555553c7 <+218>:   jmp    0x55555555536d <phase_3+128>
   0x00005555555553c9 <+220>:   callq  0x555555554ea0 <__stack_chk_fail@plt>
End of assembler dump.
```

Eax와 0x1와 cmp하고 jle하면 폭탄이 터지는 것으로 1인 것을 알 수 있다.

+88부터 +123까지 eax에 저장하는 값을 return하는 것으로 보아 -0x1b5(-437) +0x15d(349) -0x38b(907) = -995이다,

그러므로 답은 첫번째 입력값 1과 -995이다.

```
201824636@CSEDell: ~/BombLab/bomb48
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sys060/201824636/BombLab/bomb48/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
For NASA, space is still a high priority.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
^C
Program received signal SIGINT, Interrupt.
0x00007ffff7af4081 in __GI___libc_read (fd=0, buf=0x555555759e60, nbytes=1024)
    at ../sysdeps/unix/sysv/linux/read.c:27
27      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000555555555a69 <explode_bomb>
2       breakpoint     keep y   0x00005555555552ed <phase_3>
(gdb) c
Continuing.
1 -995

Breakpoint 2, 0x00005555555552ed in phase_3 ()
(gdb) c
Continuing.
Halfway there!
^C
Program received signal SIGINT, Interrupt.
0x00007ffff7af4081 in __GI___libc_read (fd=0, buf=0x555555759e60, nbytes=1024)
    at ../sysdeps/unix/sysv/linux/read.c:27
27      in ../sysdeps/unix/sysv/linux/read.c
(gdb) b phase_4
Breakpoint 3 at 0x555555555540d
```

Phase_4

```
Breakpoint 3, 0x000055555555540d in phase_4 ()
(gdb) disas phase_4
Dump of assembler code for function phase_4:
=> 0x000055555555540d <+0>:     sub    $0x18,%rsp
   0x0000555555555411 <+4>:     mov    %fs:0x28,%rax
   0x000055555555541a <+13>:    mov    %rax,0x8(%rsp)
   0x000055555555541f <+18>:    xor    %eax,%eax
   0x0000555555555421 <+20>:    lea    0x4(%rsp),%rcx
   0x0000555555555426 <+25>:    mov    %rsp,%rdx
   0x0000555555555429 <+28>:    lea    0x194d(%rip),%rsi        # 0x555555556d7d
   0x0000555555555430 <+35>:    callq  0x555555554f40 <__isoc99_sscanf@plt>
   0x0000555555555435 <+40>:    cmp    $0x2,%eax
   0x0000555555555438 <+43>:    jne    0x555555555440 <phase_4+51>
   0x000055555555543a <+45>:    cmpl   $0xe,(%rsp)
   0x000055555555543e <+49>:    jbe    0x555555555445 <phase_4+56>
   0x0000555555555440 <+51>:    callq  0x555555555a69 <explode_bomb>
   0x0000555555555445 <+56>:    mov    $0xe,%edx
   0x000055555555544a <+61>:    mov    $0x0,%esi
   0x000055555555544f <+66>:    mov    (%rsp),%edi
   0x0000555555555452 <+69>:    callq  0x5555555553ce <func4>
   0x0000555555555457 <+74>:    cmp    $0x4,%eax
   0x000055555555545a <+77>:    jne    0x555555555463 <phase_4+86>
   0x000055555555545c <+79>:    cmpl   $0x4,0x4(%rsp)
   0x0000555555555461 <+84>:    je     0x555555555468 <phase_4+91>
   0x0000555555555463 <+86>:    callq  0x555555555a69 <explode_bomb>
   0x0000555555555468 <+91>:    mov    0x8(%rsp),%rax
   0x000055555555546d <+96>:    xor    %fs:0x28,%rax
   0x0000555555555476 <+105>:   jne    0x55555555547d <phase_4+112>
   0x0000555555555478 <+107>:   add    $0x18,%rsp
   0x000055555555547c <+111>:   retq
   0x000055555555547d <+112>:   callq  0x555555554ea0 <__stack_chk_fail@plt>
End of assembler dump.
(gdb)
```

#40 eax에 들어있는 값이 2가 아니면 터짐

#45 %rsp에 들어있는 값은 0xe(15)보다 작거나 같음

%rsp에 두번째 숫자 저장됨

그러므로 답은 2, (0~15)꼴로 조합을 하여 입력하다보면

2 4 의 답을 얻을 수 있다.

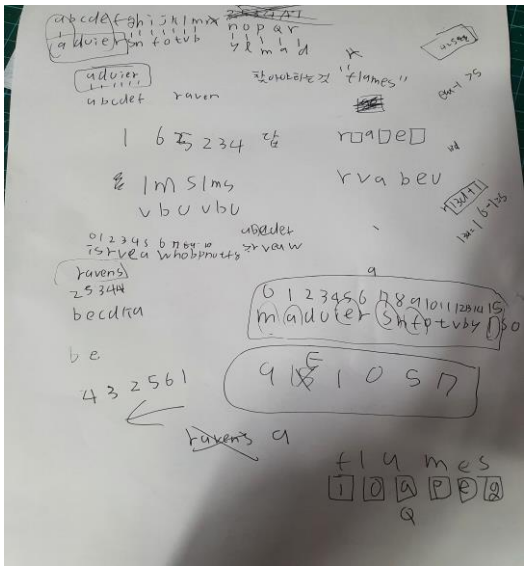
```
   0x000055555555547d <+112>:   callq  0x555555554ea0 <__stack_chk_fail@plt>
End of assembler dump.
(gdb) x/s 0x555555556d7d
0x555555556d7d: "%d %d"
(gdb)
```

```
(gdb) disas phase_5
Dump of assembler code for function phase_5:
   0x0000555555555482 <+0>:     push   %rbx
   0x0000555555555483 <+1>:     sub    $0x10,%rsp
   0x0000555555555487 <+5>:     mov    %rdi,%rbx
   0x000055555555548a <+8>:     mov    %fs:0x28,%rax
   0x0000555555555493 <+17>:    mov    %rax,0x8(%rsp)
   0x0000555555555498 <+22>:    xor    %eax,%eax
   0x000055555555549a <+24>:    callq  0x555555555748 <string_length>
   0x000055555555549f <+29>:    cmp    $0x6,%eax
   0x00005555555554a2 <+32>:    jne    0x5555555554f9 <phase_5+119>
   0x00005555555554a4 <+34>:    mov    $0x0,%eax
   0x00005555555554a9 <+39>:    lea    0x1670(%rip),%rcx        # 0x555555556b20 <array.3417>
   0x00005555555554b0 <+46>:    movzbl (%rbx,%rax,1),%edx
   0x00005555555554b4 <+50>:    and    $0xf,%edx
   0x00005555555554b7 <+53>:    movzbl (%rcx,%rdx,1),%edx
   0x00005555555554bb <+57>:    mov    %dl,0x1(%rsp,%rax,1)
   0x00005555555554bf <+61>:    add    $0x1,%rax
   0x00005555555554c3 <+65>:    cmp    $0x6,%rax
   0x00005555555554c7 <+69>:    jne    0x5555555554b0 <phase_5+46>
   0x00005555555554c9 <+71>:    movb   $0x0,0x7(%rsp)
   0x00005555555554ce <+76>:    lea    0x1(%rsp),%rdi
   0x00005555555554d3 <+81>:    lea    0x161c(%rip),%rsi        # 0x555555556af6
   0x00005555555554da <+88>:    callq  0x555555555765 <strings_not_equal>
   0x00005555555554df <+93>:    test   %eax,%eax
   0x00005555555554e1 <+95>:    jne    0x5555555555500 <phase_5+126>
   0x00005555555554e3 <+97>:    mov    0x8(%rsp),%rax
   0x00005555555554e8 <+102>:   xor    %fs:0x28,%rax
   0x00005555555554f1 <+111>:   jne    0x5555555555507 <phase_5+133>
   0x00005555555554f3 <+113>:   add    $0x10,%rsp
   0x00005555555554f7 <+117>:   pop    %rbx
   0x00005555555554f8 <+118>:   retq
   0x00005555555554f9 <+119>:   callq  0x5555555555a69 <explode_bomb>
   0x00005555555554fe <+124>:   jmp    0x5555555554a4 <phase_5+34>
   0x0000555555555500 <+126>:   callq  0x5555555555a69 <explode_bomb>
   0x0000555555555505 <+131>:   jmp    0x5555555554e3 <phase_5+97>
   0x0000555555555507 <+133>:   callq  0x55555555ea0 <__stack_chk_fail@plt>
End of assembler dump.
(gdb)
```

```
Undefined command: "0x00005555555554e8".  Try "help".
(gdb)    0x00005555555554f1 <+111>:   jne    0x5555555555507 <phase_5+133>
Undefined command: "0x00005555555554f1".  Try "help".
(gdb) x/s 0x1+$rsp
0x7fffffffe461: "aduier"
(gdb) x/s 0x161c+$rip
0x555555556af6: "flames"
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/sys060/201824636/BombLab/bomb48/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
For NASA, space is still a high priority.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
1 -995
Halfway there!
2 4
So you got that one.  Try this one.
lmslms

Breakpoint 2, 0x0000555555555482 in phase_5 ()
(gdb) c
Continuing.

Breakpoint 4, 0x00005555555554da in phase_5 ()
(gdb) x/s 0x1+$rsp
0x7fffffffe461: "vbuvbu"
(gdb)
```



phase_5:

+29에 보면 string_length함수 호출 후에 eax와 6을 비교하는 것으로 볼 때 길이가 6인 문자열을 입력받는 것으로 유추 할 수 있다.

그 후 lea연산을 하는 +76과 +81의 저장값을 조회해보면 입력한 값이 변환되어있는 string과 비교해서+95에서 not equal이면 +126의 bomb으로 날려버린다.

고로 정답은 입력값의 변환 값이 flames이면 된다.

아래 사진처럼 알파벳 순으로 입력받으면서 flames를 표현하면 정답을 찾을 수있다

답: ioapeg (flames)

Phase_6

```
(gdb) disas phase_6
Dump of assembler code for function phase_6:
   0x000055555555550c <+0>:     push   %r13
   0x000055555555550e <+2>:     push   %r12
   0x0000555555555510 <+4>:     push   %rbp
   0x0000555555555511 <+5>:     push   %rbx
   0x0000555555555512 <+6>:     sub    $0x68,%rsp
   0x0000555555555516 <+10>:    mov    %fs:0x28,%rax
   0x000055555555551f <+19>:    mov    %rax,0x58(%rsp)
   0x0000555555555524 <+24>:    xor    %eax,%eax
   0x0000555555555526 <+26>:    mov    %rsp,%r12
   0x0000555555555529 <+29>:    mov    %r12,%rsi
   0x000055555555552c <+32>:    callq  0x555555555aa5 <read_six_numbers>
   0x0000555555555531 <+37>:    mov    $0x0,%r13d
   0x0000555555555537 <+43>:    jmp    0x55555555555e <phase_6+82>
   0x0000555555555539 <+45>:    callq  0x555555555a69 <explode_bomb>
   0x000055555555553e <+50>:    jmp    0x55555555556d <phase_6+97>
   0x0000555555555540 <+52>:    add    $0x1,%ebx
   0x0000555555555543 <+55>:    cmp    $0x5,%ebx
   0x0000555555555546 <+58>:    jg     0x55555555555a <phase_6+78>
   0x0000555555555548 <+60>:    movslq %ebx,%rax
   0x000055555555554b <+63>:    mov    (%rsp,%rax,4),%eax
   0x000055555555554e <+66>:    cmp    %eax,0x0(%rbp)
   0x0000555555555551 <+69>:    jne    0x555555555540 <phase_6+52>
---Type <return> to continue, or q <return> to quit---
```

Read_six_numbers함수를 통해 6개의 input을 받는다는 사실을 알 수 있고,

```
--Type <return> to continue, or q <return> to quit---
   0x00005555555555f4 <+232>:   mov    0x8(%rbx),%rbx
   0x00005555555555f8 <+236>:   sub    $0x1,%ebp
   0x00005555555555fb <+239>:   je     0x55555555560e <phase_6+258>
   0x00005555555555fd <+241>:   mov    0x8(%rbx),%rax
   0x0000555555555601 <+245>:   mov    (%rax),%eax
   0x0000555555555603 <+247>:   cmp    %eax,(%rbx)
   0x0000555555555605 <+249>:   jle    0x5555555555f4 <phase_6+232>
   0x0000555555555607 <+251>:   callq  0x555555555a69 <explode_bomb>
   0x000055555555560c <+256>:   jmp    0x5555555555f4 <phase_6+232>
   0x000055555555560e <+258>:   mov    0x58(%rsp),%rax
   0x0000555555555613 <+263>:   xor    %fs:0x28,%rax
   0x000055555555561c <+272>:   jne    0x555555555629 <phase_6+285>
   0x000055555555561e <+274>:   add    $0x68,%rsp
   0x0000555555555622 <+278>:   pop    %rbx
   0x0000555555555623 <+279>:   pop    %rbp
   0x0000555555555624 <+280>:   pop    %r12
   0x0000555555555626 <+282>:   pop    %r13
   0x0000555555555628 <+284>:   retq
   0x0000555555555629 <+285>:   callq  0x555555554ea0 <__stack_chk_fail@plt>
nd of assembler dump.
```

+249번을 보면 cmp한 요소들이 작아야만 explode_bomb으로 가지않고 비교연산으로 돌아간다. 그러므로 오름차순 정렬로 리스트

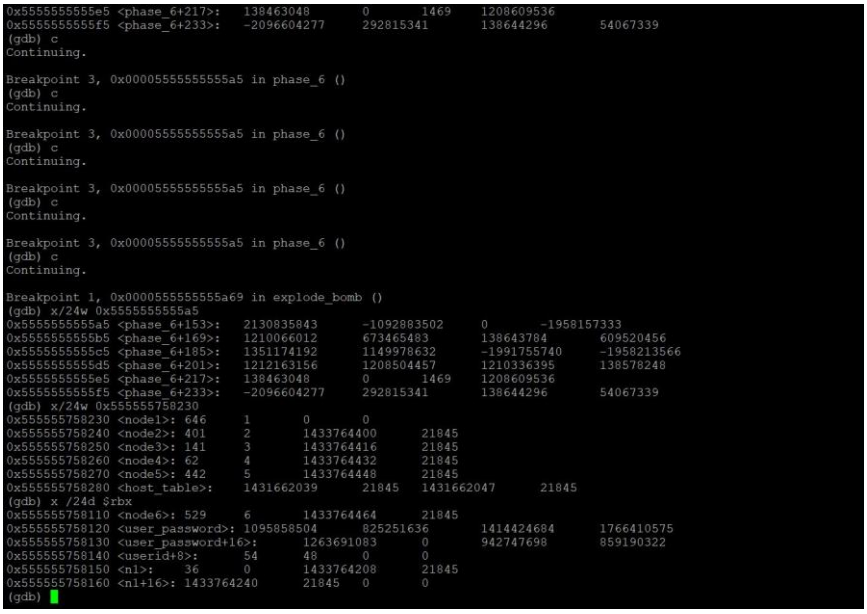
```
   0x0000555555555553 <+71>:    callq  0x555555555a69 <explode_bomb>
   0x0000555555555558 <+76>:    jmp    0x555555555540 <phase_6+52>
   0x000055555555555a <+78>:    add    $0x4,%r12
   0x000055555555555e <+82>:    mov    %r12,%rbp
   0x0000555555555561 <+85>:    mov    (%r12),%eax
   0x0000555555555565 <+89>:    sub    $0x1,%eax
   0x0000555555555568 <+92>:    cmp    $0x5,%eax
   0x000055555555556b <+95>:    ja     0x555555555539 <phase_6+45>
   0x000055555555556d <+97>:    add    $0x1,%r13d
   0x0000555555555571 <+101>:   cmp    $0x6,%r13d
   0x0000555555555575 <+105>:   je     0x5555555555ac <phase_6+160>
   0x0000555555555577 <+107>:   mov    %r13d,%ebx
   0x000055555555557a <+110>:   jmp    0x555555555548 <phase_6+60>
   0x000055555555557c <+112>:   mov    0x8(%rdx),%rdx
   0x0000555555555580 <+116>:   add    $0x1,%eax
   0x0000555555555583 <+119>:   cmp    %ecx,%eax
   0x0000555555555585 <+121>:   jne    0x55555555557c <phase_6+112>
   0x0000555555555587 <+123>:   mov    %rdx,0x20(%rsp,%rsi,8)
   0x000055555555558c <+128>:   add    $0x1,%rsi
   0x0000555555555590 <+132>:   cmp    $0x6,%rsi
   0x0000555555555594 <+136>:   je     0x5555555555b3 <phase_6+167>
   0x0000555555555596 <+138>:   mov    (%rsp,%rsi,4),%ecx
   0x0000555555555599 <+141>:   mov    $0x1,%eax
   0x000055555555559e <+146>:   lea    0x202c8b(%rip),%rdx        # 0x555555758230 <node1>
   0x00005555555555a5 <+153>:   cmp    $0x1,%ecx
   0x00005555555555a8 <+156>:   jg     0x55555555557c <phase_6+112>
   0x00005555555555aa <+158>:   jmp    0x555555555587 <phase_6+123>
   0x00005555555555ac <+160>:   mov    $0x0,%esi
   0x00005555555555b1 <+165>:   jmp    0x555555555596 <phase_6+138>
   0x00005555555555b3 <+167>:   mov    0x20(%rsp),%rbx
   0x00005555555555b8 <+172>:   mov    0x28(%rsp),%rax
   0x00005555555555bd <+177>:   mov    %rax,0x8(%rbx)
   0x00005555555555c1 <+181>:   mov    0x30(%rsp),%rdx
   0x00005555555555c6 <+186>:   mov    %rdx,0x8(%rax)
   0x00005555555555ca <+190>:   mov    0x38(%rsp),%rax
   0x00005555555555cf <+195>:   mov    %rax,0x8(%rdx)
   0x00005555555555d3 <+199>:   mov    0x40(%rsp),%rdx
   0x00005555555555d8 <+204>:   mov    %rdx,0x8(%rax)
   0x00005555555555dc <+208>:   mov    0x48(%rsp),%rax
   0x00005555555555e1 <+213>:   mov    %rax,0x8(%rdx)
   0x00005555555555e5 <+217>:   movq   $0x0,0x8(%rax)
   0x00005555555555ed <+225>:   mov    $0x5,%ebp
   0x00005555555555f2 <+230>:   jmp    0x5555555555fd <phase_6+241>
```

#에 첨부되어있는 주소값을 통해 node의 값들을 알 수있고,

+89와 +92에서 node의 값들이

eax-1<=5

즉 eax는 6 이하여야 한다는 사실을 알 수 있다.



첨부되어 있는 노드들의 값을 내림차순으로 정렬하여 입력해주면 정답.