```
Breakpoint 2, 0x00005555555558b2 in phase_6 ()
(gdb) disas
Dump of assembler code for function phase_6:
=> 0x00005555555558b2 <+0>:     endbr64
   0x00005555555558b6 <+4>:     push   %r14
   0x00005555555558b8 <+6>:     push   %r13
   0x00005555555558ba <+8>:     push   %r12
   0x00005555555558bc <+10>:    push   %rbp
   0x00005555555558bd <+11>:    push   %rbx
   0x00005555555558be <+12>:    sub    $0x60,%rsp
   0x00005555555558c2 <+16>:    mov    %fs:0x28,%rax
   0x00005555555558cb <+25>:    mov    %rax,0x58(%rsp)
   0x00005555555558d0 <+30>:    xor    %eax,%eax
   0x00005555555558d2 <+32>:    mov    %rsp,%r13
   0x00005555555558d5 <+35>:    mov    %r13,%rsi
   0x00005555555558d8 <+38>:    call   0x5555555555e51 <read_six_numbers>
   0x00005555555558dd <+43>:    mov    $0x1,%r14d
   0x00005555555558e3 <+49>:    mov    %rsp,%r12
   0x00005555555558e6 <+52>:    jmp    0x5555555559l0 <phase_6+94>
   0x00005555555558e8 <+54>:    call   0x5555555555e0f <explode_bomb>
   0x00005555555558ed <+59>:    jmp    0x5555555591f <phase_6+109>
   0x00005555555558ef <+61>:    add    $0x1,%rbx
   0x00005555555558f3 <+65>:    cmp    $0x5,%ebx
   0x00005555555558f6 <+68>:    jg     0x5555555559l8 <phase_6+86>
   0x00005555555558f8 <+70>:    mov    (%r12,%rbx,4),%eax
--Type <RET> for more, q to quit, c to continue without paging--
   0x00005555555558fc <+74>:    cmp    %eax,0x0(%rbp)
   0x00005555555558ff <+77>:    jne    0x5555555558ef <phase_6+61>
   0x0000555555555901 <+79>:    call   0x5555555555e0f <explode_bomb>
   0x0000555555555906 <+84>:    jmp    0x5555555558ef <phase_6+61>
   0x0000555555555908 <+86>:    add    $0x1,%r14
   0x000055555555590c <+90>:    add    $0x4,%r13
   0x0000555555555910 <+94>:    mov    %r13,%rbp
   0x0000555555555913 <+97>:    mov    0x0(%r13),%eax
   0x0000555555555917 <+101>:   sub    $0x1,%eax
   0x000055555555591a <+104>:   cmp    $0x5,%eax
   0x000055555555591d <+107>:   ja     0x5555555558e8 <phase_6+54>
   0x000055555555591f <+109>:   cmp    $0x5,%r14d
   0x0000555555555923 <+113>:   jg     0x555555555592a <phase_6+120>
   0x0000555555555925 <+115>:   mov    %r14,%rbx
   0x0000555555555928 <+118>:   jmp    0x5555555558f8 <phase_6+70>
   0x000055555555592a <+120>:   mov    $0x0,%esi
   0x000055555555592f <+125>:   mov    (%rsp,%rsi,4),%ecx
   0x0000555555555932 <+128>:   mov    $0x1,%eax
   0x0000555555555937 <+133>:   lea    0x3cf2(%rip),%rdx        # 0x555555559630 <node1>
   0x000055555555593e <+140>:   cmp    $0x1,%ecx
   0x0000555555555941 <+143>:   jle    0x555555555594e <phase_6+156>
   0x0000555555555943 <+145>:   mov    0x8(%rdx),%rdx
--Type <RET> for more, q to quit, c to continue without paging--
   0x0000555555555947 <+149>:   add    $0x1,%eax
   0x000055555555594a <+152>:   cmp    %ecx,%eax
   0x000055555555594c <+154>:   jne    0x5555555559l3 <phase_6+145>
   0x000055555555594e <+156>:   mov    %rdx,0x20(%rsp,%rsi,8)
   0x0000555555555953 <+161>:   add    $0x1,%rsi
   0x0000555555555957 <+165>:   cmp    $0x6,%rsi
   0x000055555555595b <+169>:   jne    0x555555555592f <phase_6+125>
   0x000055555555595d <+171>:   mov    0x20(%rsp),%rbx
   0x0000555555555962 <+176>:   mov    0x28(%rsp),%rax
   0x0000555555555967 <+181>:   mov    %rax,0x8(%rbx)
   0x000055555555596b <+185>:   mov    0x30(%rsp),%rdx
   0x0000555555555970 <+190>:   mov    %rdx,0x8(%rax)
   0x0000555555555974 <+194>:   mov    0x38(%rsp),%rax
   0x0000555555555979 <+199>:   mov    %rax,0x8(%rdx)

   0x0000555555555982 <+208>:   mov    %rdx,0x8(%rax)
   0x0000555555555986 <+212>:   mov    0x48(%rsp),%rax
   0x000055555555598b <+217>:   mov    %rax,0x8(%rdx)
   0x000055555555598f <+221>:   movq   $0x0,0x8(%rax)
   0x0000555555555997 <+229>:   mov    $0x5,%ebp
   0x000055555555599c <+234>:   jmp    0x5555555559a7 <phase_6+245>
   0x000055555555599e <+236>:   mov    0x8(%rbx),%rbx
   0x00005555555559a2 <+240>:   sub    $0x1,%ebp
--Type <RET> for more, q to quit, c to continue without paging--
   0x00005555555559a5 <+243>:   je     0x5555555559b8 <phase_6+262>
   0x00005555555559a7 <+245>:   mov    0x8(%rbx),%rax
   0x00005555555559ab <+249>:   mov    (%rax),%eax
   0x00005555555559ad <+251>:   cmp    %eax,(%rbx)
   0x00005555555559af <+253>:   jge    0x5555555559e <phase_6+236>
   0x00005555555559b1 <+255>:   call   0x5555555555e0f <explode_bomb>
   0x00005555555559b6 <+260>:   jmp    0x5555555559e <phase_6+236>
   0x00005555555559b8 <+262>:   mov    0x58(%rsp),%rax
   0x00005555555559bd <+267>:   sub    %fs:0x28,%rax
   0x00005555555559c6 <+276>:   jne    0x5555555559d5 <phase_6+291>
   0x00005555555559c8 <+278>:   add    $0x60,%rsp
   0x00005555555559cc <+282>:   pop    %rbx
   0x00005555555559cd <+283>:   pop    %rbp
   0x00005555555559ce <+284>:   pop    %r12
   0x00005555555559d0 <+286>:   pop    %r13
   0x00005555555559d2 <+288>:   pop    %r14
   0x00005555555559d4 <+290>:   ret
   0x00005555555559d5 <+291>:   call   0x5555555555280 <__stack_chk_fail@plt>
End of assembler dump.
(gdb)
```

Ex: (1 2 4 16 8 9)

← Read six numbers

moves 1 into ojer14d

moves ojersp's value into ojer12

← adds 1 to ojeebx (our input numbers)

checks for any numbers that match! (we can't have any numbers that match!)

← moves current value to an index

← compares the current value with ojeeax's value

ojerbp → possibly our input?

← mov ojer13's value into ojerbp

← mov ojer13's value into ojeeax (check input) → with our ex input: ojerax = 1 (our first number)

← subtract 1 from ojeeax (so 1-1 = 0 for ex input)

← compares 5 with ojeeax

(Do not use any number above 5!)

(our new input: 1 2 3 4 5 6)

If ojeeax > 5 (go to explode bomb)

checks all 6 numbers for if each number is > 5

← mov 0 to ojeesi

ojeecx = memory address of ojersp + 4·ojersi ← moves our first element into ojeecx = 1

← mov 1 to ojeeax

← If ojeecx <= 1

This will increment and write 6 values

← when ojeecx > 1, we look at the address of ojerdx (node1) and adds 8 to the address (possibly for a counter)

← when the value we gave it (our input) matches ojeeax, then we will move that value to the stack

(ojersp + 8·ojersi + 0x20)

=0

ojerdx is currently holding node1 → gets sent to node1 → (ojersp + 0x20)

Iterating 6 times

```
(gdb) x/d 0x555555559630
0x555555559630 <node1>: 145
(gdb) x/1gx 0x555555559630
0x555555559630 <node1>: 0x0000000100000091
(gdb) x/6gx 0x555555559630
0x555555559630 <node1>: 0x0000000100000091      0x0000555555559640
0x555555559640 <node2>: 0x000000020000012d      0x0000555555559650
0x555555559650 <node3>: 0x00000003000001bb      0x0000555555559660
(gdb) x/12gx 0x555555559630
0x555555559630 <node1>: 0x0000000100000091      0x0000555555559640
0x555555559640 <node2>: 0x000000020000012d      0x0000555555559650
0x555555559650 <node3>: 0x00000003000001bb      0x0000555555559660
0x555555559660 <node4>: 0x000000043000003c      0x0000555555559670
0x555555559670 <node5>: 0x0000000500000001      0x0000555555559120
0x555555559680 <host_table>:    0x0000555555557745f     0x0000555555557470
(gdb) x/2gx 0x0000555555559120
0x555555559120 <node6>: 0x0000000600000034      0x0000000000000000
(gdb)
```

	num	int (4 byte)	Address
Node 1	1 ← 2nd	91	" 30
Node 2	2 ← 3rd	12d	0x0000555555559640
Node 3	3 ← 5th	166	" 50
Node 4	4 ← 1st	3C	" 60
Node 5	5 ← 4th	1AF	" 70
Node 6	6 ← 6th	34a	" 120
		635214	0x0000...

Now... at <t253>

Round 1

443 (%eax) 842 ((%r6x))

If ((%r6x)) ≥ %eax
Don't explode

cmp %eax, ((%r6x))
jge

Round 2:

eax = 431 ((%r6x)) = 443

Round 3:

eax = 301 ((%r6x)) = 431

Round 4:

eax = 145 ((%r6x)) = 301

Round 5:

eax = 60 (3c) ((%r6x)) = 145

Round 6:

eax = 431 ((%r6x)) = 60