Dump of assembler code for function phase\_6:

0x00000000004010a0 <+0>: push %r13

0x00000000004010a2 <+2>: push %r12

0x00000000004010a4 <+4>: push %rbp

0x00000000004010a5 <+5>: push %rbx

0x00000000004010a6 <+6>: sub $0x58,%rsp

0x00000000004010aa <+10>: lea 0x30(%rsp),%rsi

0x00000000004010af <+15>: callq 0x4014bc <read\_six\_numbers> //读6个数字

0x00000000004010b4 <+20>: lea 0x30(%rsp),%r12

0x00000000004010b9 <+25>: mov $0x0,%r13d

0x00000000004010bf <+31>: mov %r12,%rbp

0x00000000004010c2 <+34>: mov (%r12),%eax

0x00000000004010c6 <+38>: sub $0x1,%eax

0x00000000004010c9 <+41>: cmp $0x5,%eax //所有数字小于等于六

0x00000000004010cc <+44>: jbe 0x4010d3 <phase\_6+51>

0x00000000004010ce <+46>: callq 0x401486 <explode\_bomb>

这个是一个while loop，判断这六个数是否等于零。推测到每个数字都不为零

0x00000000004010d3 <+51>: add $0x1,%r13d

0x00000000004010d7 <+55>: cmp $0x6,%r13d

0x00000000004010db <+59>: je 0x40111a <phase\_6+122>

0x00000000004010dd <+61>: mov %r13d,%ebx

0x00000000004010e0 <+64>: movslq %ebx,%rax

0x00000000004010e3 <+67>: mov 0x30(%rsp,%rax,4),%eax

0x00000000004010e7 <+71>: cmp %eax,0x0(%rbp) //判断每个数字都不为零

0x00000000004010ea <+74>: jne 0x4010f1 <phase\_6+81>

0x00000000004010ec <+76>: callq 0x401486 <explode\_bomb>

0x00000000004010f1 <+81>: add $0x1,%ebx

0x00000000004010f4 <+84>: cmp $0x5,%ebx

0x00000000004010f7 <+87>: jle 0x4010e0 <phase\_6+64>

0x00000000004010f9 <+89>: add $0x4,%r12

0x00000000004010fd <+93>: jmp 0x4010bf <phase\_6+31>

0x00000000004010ff <+95>: mov 0x8(%rdx),%rdx

0x0000000000401103 <+99>: add $0x1,%eax

0x0000000000401106 <+102>: cmp %ecx,%eax

0x0000000000401108 <+104>: jne 0x4010ff <phase\_6+95>

0x000000000040110a <+106>: mov %rdx,(%rsp,%rsi,2)

0x000000000040110e <+110>: add $0x4,%rsi

0x0000000000401112 <+114>: cmp $0x18,%rsi

0x0000000000401116 <+118>: jne 0x40111f <phase\_6+127>

0x0000000000401118 <+120>: jmp 0x401134 <phase\_6+148>

0x000000000040111a <+122>: mov $0x0,%esi

0x000000000040111f <+127>: mov 0x30(%rsp,%rsi,1),%ecx

0x0000000000401123 <+131>: mov $0x1,%eax

0x0000000000401128 <+136>: mov $0x6042e0,%edx

0x000000000040112d <+141>: cmp $0x1,%ecx

0x0000000000401130 <+144>: jg 0x4010ff <phase\_6+95>

0x0000000000401132 <+146>: jmp 0x40110a <phase\_6+106>

0x0000000000401134 <+148>: mov (%rsp),%rbx

0x0000000000401138 <+152>: mov %rsp,%rax

0x000000000040113b <+155>: lea 0x28(%rsp),%rsi

0x0000000000401140 <+160>: mov %rbx,%rcx

0x0000000000401143 <+163>: mov 0x8(%rax),%rdx

0x0000000000401147 <+167>: mov %rdx,0x8(%rcx)//将下一个NODE的地址赋值给

0x000000000040114b <+171>: add $0x8,%rax

0x000000000040114f <+175>: mov %rdx,%rcx

0x0000000000401152 <+178>: cmp %rsi,%rax

0x0000000000401155 <+181>: jne 0x401143 <phase\_6+163>

0x0000000000401157 <+183>: movq $0x0,0x8(%rdx) //最后的指针为NULL

下面代码用来判断链表是否递增，如果递减，则不爆炸，否则爆炸

0x000000000040115f <+191>: mov $0x5,%ebp

0x0000000000401164 <+196>: mov 0x8(%rbx),%rax

0x0000000000401168 <+200>: mov (%rax),%eax

0x000000000040116a <+202>: cmp %eax,(%rbx) //前一个是否大于后一个

0x000000000040116c <+204>: jge 0x401173 <phase\_6+211>

0x000000000040116e <+206>: callq 0x401486 <explode\_bomb>

0x0000000000401173 <+211>: mov 0x8(%rbx),%rbx

0x0000000000401177 <+215>: sub $0x1,%ebp

0x000000000040117a <+218>: jne 0x401164 <phase\_6+196>

0x000000000040117c <+220>: add $0x58,%rsp

0x0000000000401180 <+224>: pop %rbx

0x0000000000401181 <+225>: pop %rbp

0x0000000000401182 <+226>: pop %r12

0x0000000000401184 <+228>: pop %r13

0x0000000000401186 <+230>: retq

End of assembler dump.

p/s \*0x6042e0@24