CA ASSIGNMENT LAB – 05

BOMB LAB

BOMB ID: 207

BOMB LAB CONSISTS 6 PHASES

In each phases we use GDB debugger and we find the logic and find the required values for each phases.

Command for using GDB is:

gdb bomb

Phase 1:

```
rahul@rahul:~/Documents/lab_bomb$ gdb bomb
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <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...
(gdb) break string_length
Breakpoint 1 at 0x401309
(gdb) run
Starting program: /home/rahul/Documents/lab_bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
hello

Breakpoint 1, 0x0000000000401309 in string_length ()
(gdb) print $rdi
$1 = 6305696
(gdb) x 6305696
0x6037a0 <input_strings>:       0x6c6c6568
(gdb) x/s 6305696
0x6037a0 <input_strings>:       "hello"
(gdb) continue
Continuing.

Breakpoint 1, 0x0000000000401309 in string_length ()
(gdb) print $rdi
$2 = 4203440
(gdb) x/s 4203440
0x4023b0:       "Houses will begat jobs, jobs will begat houses."
(gdb)
```

Phase 1 process:

By analyzing code we can say that required answer is string

1. Use GDB bomb.

2. Break at string_length.

3. Run the program and give input as hello.

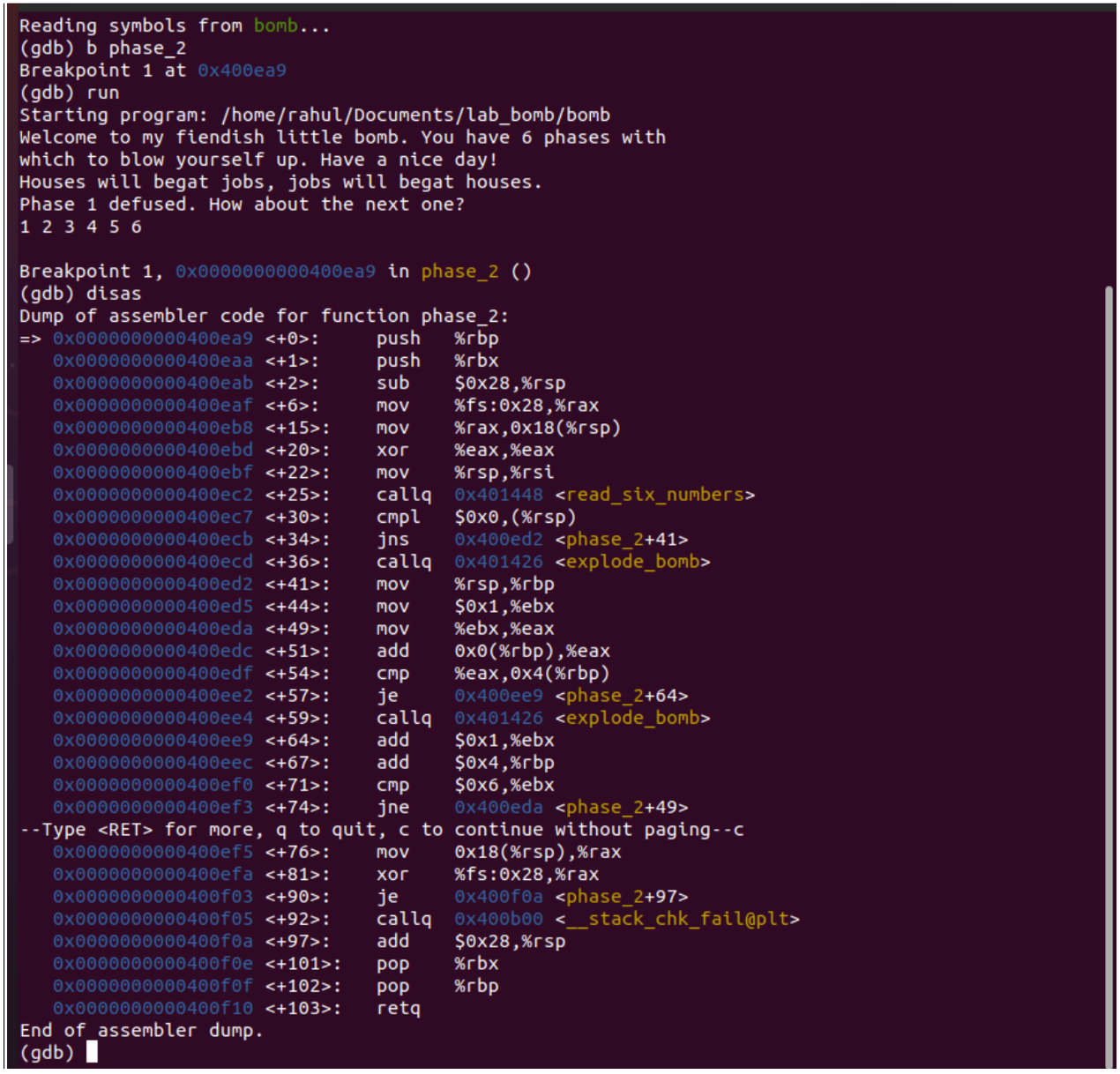
4. Print $rdi and use x/s for the value, we get hello.

5. Now at another break point we get the required value.

Final value in phase 1 is :

“Houses will begat jobs, jobs will begat houses.”

Phase 2:



```
Reading symbols from bomb...
(gdb) b phase_2
Breakpoint 1 at 0x400ea9
(gdb) run
Starting program: /home/rahul/Documents/lab_bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Houses will begat jobs, jobs will begat houses.
Phase 1 defused. How about the next one?
1 2 3 4 5 6

Breakpoint 1, 0x0000000000400ea9 in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
=> 0x0000000000400ea9 <+0>:     push   %rbp
   0x0000000000400eaa <+1>:     push   %rbx
   0x0000000000400eab <+2>:     sub    $0x28,%rsp
   0x0000000000400eaf <+6>:     mov    %fs:0x28,%rax
   0x0000000000400eb8 <+15>:    mov    %rax,0x18(%rsp)
   0x0000000000400ebd <+20>:    xor    %eax,%eax
   0x0000000000400ebf <+22>:    mov    %rsp,%rsi
   0x0000000000400ec2 <+25>:    callq  0x401448 <read_six_numbers>
   0x0000000000400ec7 <+30>:    cmpl   $0x0,(%rsp)
   0x0000000000400ecb <+34>:    jns    0x400ed2 <phase_2+41>
   0x0000000000400ecd <+36>:    callq  0x401426 <explode_bomb>
   0x0000000000400ed2 <+41>:    mov    %rsp,%rbp
   0x0000000000400ed5 <+44>:    mov    $0x1,%ebx
   0x0000000000400eda <+49>:    mov    %ebx,%eax
   0x0000000000400edc <+51>:    add    0x0(%rbp),%eax
   0x0000000000400edf <+54>:    cmp    %eax,0x4(%rbp)
   0x0000000000400ee2 <+57>:    je     0x400ee9 <phase_2+64>
   0x0000000000400ee4 <+59>:    callq  0x401426 <explode_bomb>
   0x0000000000400ee9 <+64>:    add    $0x1,%ebx
   0x0000000000400eec <+67>:    add    $0x4,%rbp
   0x0000000000400ef0 <+71>:    cmp    $0x6,%ebx
   0x0000000000400ef3 <+74>:    jne    0x400eda <phase_2+49>
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000000000400ef5 <+76>:    mov    0x18(%rsp),%rax
   0x0000000000400efa <+81>:    xor    %fs:0x28,%rax
   0x0000000000400f03 <+90>:    je     0x400f0a <phase_2+97>
   0x0000000000400f05 <+92>:    callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400f0a <+97>:    add    $0x28,%rsp
   0x0000000000400f0e <+101>:   pop    %rbx
   0x0000000000400f0f <+102>:   pop    %rbp
   0x0000000000400f10 <+103>:   retq
End of assembler dump.
(gdb)
```

```
Breakpoint 1, 0x0000000000401448 in read_six_numbers ()
(gdb) disas
Dump of assembler code for function read_six_numbers:
=> 0x0000000000401448 <+0>:     sub    $0x8,%rsp
   0x000000000040144c <+4>:     mov    %rsi,%rdx
   0x000000000040144f <+7>:     lea    0x4(%rsi),%rcx
   0x0000000000401453 <+11>:    lea    0x14(%rsi),%rax
   0x0000000000401457 <+15>:    push   %rax
   0x0000000000401458 <+16>:    lea    0x10(%rsi),%rax
   0x000000000040145c <+20>:    push   %rax
   0x000000000040145d <+21>:    lea    0xc(%rsi),%r9
   0x0000000000401461 <+25>:    lea    0x8(%rsi),%r8
   0x0000000000401465 <+29>:    mov    $0x4025a3,%esi
   0x000000000040146a <+34>:    mov    $0x0,%eax
   0x000000000040146f <+39>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x0000000000401474 <+44>:    add    $0x10,%rsp
   0x0000000000401478 <+48>:    cmp    $0x5,%eax
   0x000000000040147b <+51>:    jg     0x401482 <read_six_numbers+58>
   0x000000000040147d <+53>:    callq  0x401426 <explode_bomb>
   0x0000000000401482 <+58>:    add    $0x8,%rsp
   0x0000000000401486 <+62>:    retq
End of assembler dump.
(gdb)
```

This phase is solved by just analyzing the code

By assembly language knowledge we can get the required answer for phase 2.

Process for phase 2:

1. By analyzing code we know that there are six numbers and the starting number is 1
2. For the next five number in a sequence some value is added
3. For second number one is added to the previous number
4. For third number two is added to the previous number
5. For fourth number three is added to the previous number
6. For fifth number four is added to the previous number
7. For last (sixth) number five is added to the previous number

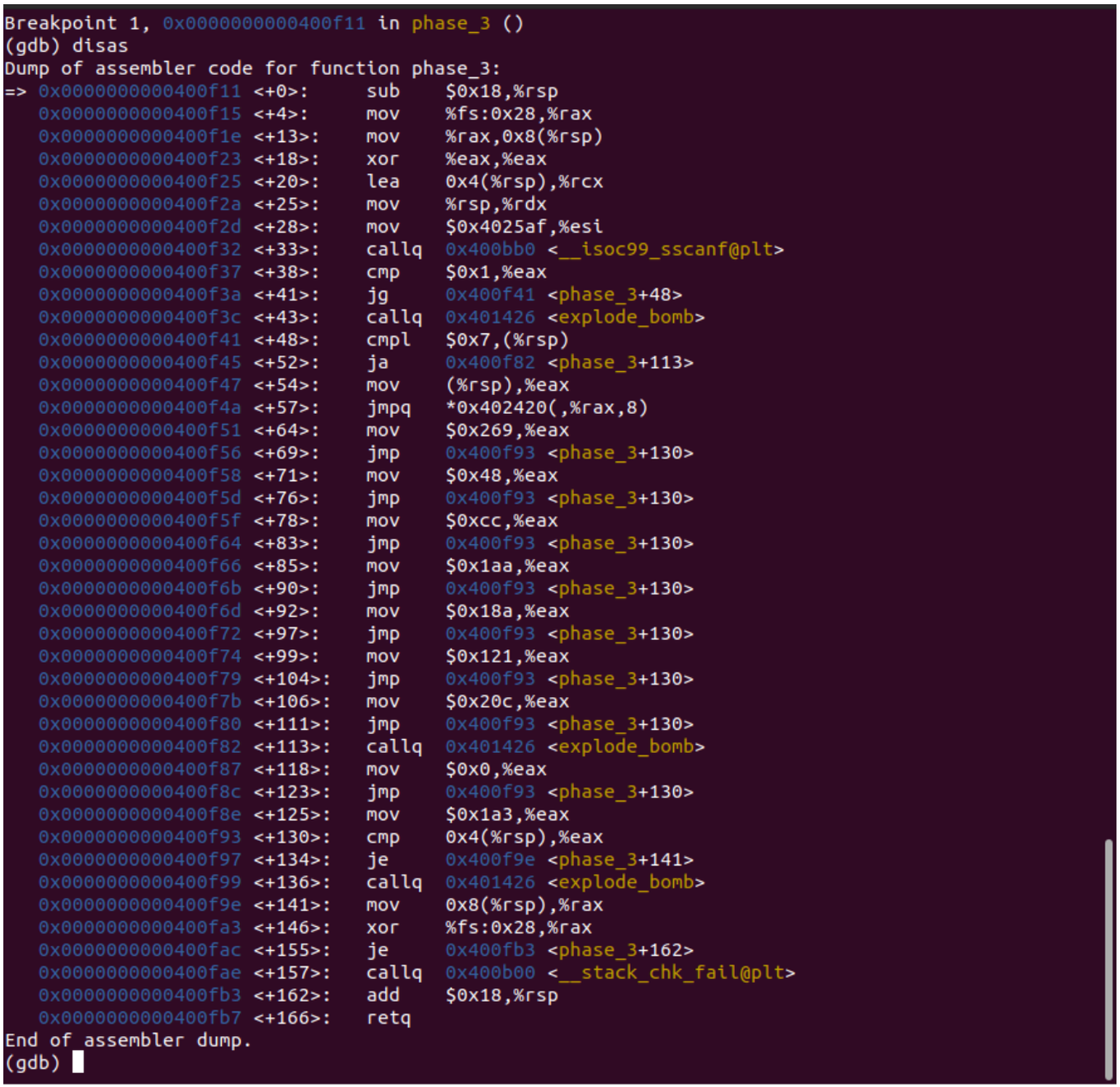
Finally the answer for phase 2:

1 2 4 7 11 16

Phase 3:

```
(gdb) b phase_3
Breakpoint 1 at 0x400f11
(gdb) run
Starting program: /home/rahul/Documents/lab_bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Houses will begat jobs, jobs will begat houses.
Phase 1 defused. How about the next one?
1 2 4 7 11 16
That's number 2.  Keep going!
1 2

Breakpoint 1, 0x0000000000400f11 in phase_3 ()
(gdb) disas
Dump of assembler code for function phase_3:
=> 0x0000000000400f11 <+0>:     sub    $0x18,%rsp
   0x0000000000400f15 <+4>:     mov    %fs:0x28,%rax
   0x0000000000400f1e <+13>:    mov    %rax,0x8(%rsp)
   0x0000000000400f23 <+18>:    xor    %eax,%eax
   0x0000000000400f25 <+20>:    lea    0x4(%rsp),%rcx
   0x0000000000400f2a <+25>:    mov    %rsp,%rdx
   0x0000000000400f2d <+28>:    mov    $0x4025af,%esi
   0x0000000000400f32 <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x0000000000400f37 <+38>:    cmp    $0x1,%eax
   0x0000000000400f3a <+41>:    jg     0x400f41 <phase_3+48>
   0x0000000000400f3c <+43>:    callq  0x401426 <explode_bomb>
   0x0000000000400f41 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f45 <+52>:    ja     0x400f82 <phase_3+113>
   0x0000000000400f47 <+54>:    mov    (%rsp),%eax
   0x0000000000400f4a <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f51 <+64>:    mov    $0x269,%eax
   0x0000000000400f56 <+69>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f58 <+71>:    mov    $0x48,%eax
   0x0000000000400f5d <+76>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f5f <+78>:    mov    $0xcc,%eax
   0x0000000000400f64 <+83>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f66 <+85>:    mov    $0x1aa,%eax
   0x0000000000400f6b <+90>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f6d <+92>:    mov    $0x18a,%eax
   0x0000000000400f72 <+97>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f74 <+99>:    mov    $0x121,%eax
   0x0000000000400f79 <+104>:   jmp    0x400f93 <phase_3+130>
   0x0000000000400f7b <+106>:   mov    $0x20c,%eax
   0x0000000000400f80 <+111>:   jmp    0x400f93 <phase_3+130>
   0x0000000000400f82 <+113>:   callq  0x401426 <explode_bomb>
   0x0000000000400f87 <+118>:   mov    $0x0,%eax
   0x0000000000400f8c <+123>:   jmp    0x400f93 <phase_3+130>
```



```
Breakpoint 1, 0x0000000000400f11 in phase_3 ()
(gdb) disas
Dump of assembler code for function phase_3:
=> 0x0000000000400f11 <+0>:     sub    $0x18,%rsp
   0x0000000000400f15 <+4>:     mov    %fs:0x28,%rax
   0x0000000000400f1e <+13>:    mov    %rax,0x8(%rsp)
   0x0000000000400f23 <+18>:    xor    %eax,%eax
   0x0000000000400f25 <+20>:    lea    0x4(%rsp),%rcx
   0x0000000000400f2a <+25>:    mov    %rsp,%rdx
   0x0000000000400f2d <+28>:    mov    $0x4025af,%esi
   0x0000000000400f32 <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x0000000000400f37 <+38>:    cmp    $0x1,%eax
   0x0000000000400f3a <+41>:    jg     0x400f41 <phase_3+48>
   0x0000000000400f3c <+43>:    callq  0x401426 <explode_bomb>
   0x0000000000400f41 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f45 <+52>:    ja     0x400f82 <phase_3+113>
   0x0000000000400f47 <+54>:    mov    (%rsp),%eax
   0x0000000000400f4a <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f51 <+64>:    mov    $0x269,%eax
   0x0000000000400f56 <+69>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f58 <+71>:    mov    $0x48,%eax
   0x0000000000400f5d <+76>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f5f <+78>:    mov    $0xcc,%eax
   0x0000000000400f64 <+83>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f66 <+85>:    mov    $0x1aa,%eax
   0x0000000000400f6b <+90>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f6d <+92>:    mov    $0x18a,%eax
   0x0000000000400f72 <+97>:    jmp    0x400f93 <phase_3+130>
   0x0000000000400f74 <+99>:    mov    $0x121,%eax
   0x0000000000400f79 <+104>:   jmp    0x400f93 <phase_3+130>
   0x0000000000400f7b <+106>:   mov    $0x20c,%eax
   0x0000000000400f80 <+111>:   jmp    0x400f93 <phase_3+130>
   0x0000000000400f82 <+113>:   callq  0x401426 <explode_bomb>
   0x0000000000400f87 <+118>:   mov    $0x0,%eax
   0x0000000000400f8c <+123>:   jmp    0x400f93 <phase_3+130>
   0x0000000000400f8e <+125>:   mov    $0x1a3,%eax
   0x0000000000400f93 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f97 <+134>:   je     0x400f9e <phase_3+141>
   0x0000000000400f99 <+136>:   callq  0x401426 <explode_bomb>
   0x0000000000400f9e <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa3 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fac <+155>:   je     0x400fb3 <phase_3+162>
   0x0000000000400fae <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb3 <+162>:   add    $0x18,%rsp
   0x0000000000400fb7 <+166>:   retq
End of assembler dump.
(gdb)
```

Process for phase 3:

1. This phase contains switch statements
2. This phase contains two integers by using x/s 0x4025af

```
(gdb) x/s 0x4025af
0x4025af:       "%d %d"
(gdb)
```

3. By seeing code we can analyze

```
0x000000000400f37 <+38>:    cmp    $0x1,%eax
0x000000000400f3a <+41>:    jg     0x400f41 <phase_3+48>
0x000000000400f3c <+43>:    callq  0x401426 <explode_bomb>
```

4. And by seeing appropriate switch case we can get the final values
5. By considering the first value as 4

and calculating the next value by seeing the above code(last page) and converting into binary we get 426

The final value for phase 3 is:

4 426

Phase 4:

```
(gdb) b phase_4
Breakpoint 1 at 0x400ff6
(gdb) run
Starting program: /home/rahul/Documents/lab_bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Houses will begat jobs, jobs will begat houses.
Phase 1 defused. How about the next one?
1 2 4 7 11 16
That's number 2.  Keep going!
4 426
Halfway there!
1 2

Breakpoint 1, 0x0000000000400ff6 in phase_4 ()
(gdb) disas
Dump of assembler code for function phase_4:
=> 0x0000000000400ff6 <+0>:     sub    $0x18,%rsp
   0x0000000000400ffa <+4>:     mov    %fs:0x28,%rax
   0x0000000000401003 <+13>:    mov    %rax,0x8(%rsp)
   0x0000000000401008 <+18>:    xor    %eax,%eax
   0x000000000040100a <+20>:    lea    0x4(%rsp),%rcx
   0x000000000040100f <+25>:    mov    %rsp,%rdx
   0x0000000000401012 <+28>:    mov    $0x4025af,%esi
   0x0000000000401017 <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x000000000040101c <+38>:    cmp    $0x2,%eax
   0x000000000040101f <+41>:    jne    0x401027 <phase_4+49>
   0x0000000000401021 <+43>:    cmpl   $0xe,(%rsp)
   0x0000000000401025 <+47>:    jbe    0x40102c <phase_4+54>
   0x0000000000401027 <+49>:    callq  0x401426 <explode_bomb>
   0x000000000040102c <+54>:    mov    $0xe,%edx
   0x0000000000401031 <+59>:    mov    $0x0,%esi
   0x0000000000401036 <+64>:    mov    (%rsp),%edi
   0x0000000000401039 <+67>:    callq  0x400fb8 <func4>
   0x000000000040103e <+72>:    cmp    $0x2,%eax
   0x0000000000401041 <+75>:    jne    0x40104a <phase_4+84>
   0x0000000000401043 <+77>:    cmpl   $0x2,0x4(%rsp)
   0x0000000000401048 <+82>:    je     0x40104f <phase_4+89>
   0x000000000040104a <+84>:    callq  0x401426 <explode_bomb>
   0x000000000040104f <+89>:    mov    0x8(%rsp),%rax
   0x0000000000401054 <+94>:    xor    %fs:0x28,%rax
   0x000000000040105d <+103>:   je     0x401064 <phase_4+110>
   0x000000000040105f <+105>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000401064 <+110>:   add    $0x18,%rsp
   0x0000000000401068 <+114>:   retq
End of assembler dump.
(gdb)
```

Process for phase 4:

1.This phase contains two integers

```
End of assembler dump.
(gdb) x/s 0x4025af
0x4025af:       "%d %d"
```

2.By the below picture we can analyze that first integer lies between 1 and 14

```
0x000000000040101c <+38>:    cmp    $0x2,%eax
0x000000000040101f <+41>:    jne    0x401027 <phase_4+49>
0x0000000000401021 <+43>:    cmpl   $0xe,(%rsp)
0x0000000000401025 <+47>:    jbe    0x40102c <phase_4+54>
0x0000000000401027 <+49>:    callq  0x401426 <explode_bomb>
```

3.By the below photo we can analyze that the second integer is 2

```
0x0000000000401043 <+77>:    cmpl   $0x2,0x4(%rsp)
0x0000000000401048 <+82>:    je     0x40104f <phase_4+89>
0x000000000040104a <+84>:    callq  0x401426 <explode_bomb>
```

4. By keeping value for first integer from 1 to 14 by keeping second integer 2, we can get the first integer as 4

The final answer for phase 4 is:

4 2

Phase 5:

```
(gdb) b phase_5
Breakpoint 1 at 0x401069
(gdb) run
Starting program: /home/rahul/Documents/lab_bomb/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Houses will begat jobs, jobs will begat houses.
Phase 1 defused. How about the next one?
1 2 4 7 11 16
That's number 2.  Keep going!
4 426
Halfway there!
4 2
So you got that one.  Try this one.
1 2

Breakpoint 1, 0x0000000000401069 in phase_5 ()
(gdb) disas
Dump of assembler code for function phase_5:
=> 0x0000000000401069 <+0>:     sub    $0x18,%rsp
   0x000000000040106d <+4>:     mov    %fs:0x28,%rax
   0x0000000000401076 <+13>:    mov    %rax,0x8(%rsp)
   0x000000000040107b <+18>:    xor    %eax,%eax
   0x000000000040107d <+20>:    lea    0x4(%rsp),%rcx
   0x0000000000401082 <+25>:    mov    %rsp,%rdx
   0x0000000000401085 <+28>:    mov    $0x4025af,%esi
   0x000000000040108a <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x000000000040108f <+38>:    cmp    $0x1,%eax
   0x0000000000401092 <+41>:    jg     0x401099 <phase_5+48>
   0x0000000000401094 <+43>:    callq  0x401426 <explode_bomb>
   0x0000000000401099 <+48>:    mov    (%rsp),%eax
   0x000000000040109c <+51>:    and    $0xf,%eax
   0x000000000040109f <+54>:    mov    %eax,(%rsp)
   0x00000000004010a2 <+57>:    cmp    $0xf,%eax
   0x00000000004010a5 <+60>:    je     0x4010d6 <phase_5+109>
   0x00000000004010a7 <+62>:    mov    $0x0,%ecx
   0x00000000004010ac <+67>:    mov    $0x0,%edx
   0x00000000004010b1 <+72>:    add    $0x1,%edx
   0x00000000004010b4 <+75>:    cltq
   0x00000000004010b6 <+77>:    mov    0x402460(,%rax,4),%eax
   0x00000000004010bd <+84>:    add    %eax,%ecx
   0x00000000004010bf <+86>:    cmp    $0xf,%eax
   0x00000000004010c2 <+89>:    jne    0x4010b1 <phase_5+72>
   0x00000000004010c4 <+91>:    movl   $0xf,(%rsp)
   0x00000000004010cb <+98>:    cmp    $0xf,%edx
   0x00000000004010ce <+101>:   jne    0x4010d6 <phase_5+109>
   0x00000000004010d0 <+103>:   cmp    0x4(%rsp),%ecx
```

```
Breakpoint 1, 0x0000000000401069 in phase_5 ()
(gdb) disas
Dump of assembler code for function phase_5:
=> 0x0000000000401069 <+0>:     sub    $0x18,%rsp
   0x000000000040106d <+4>:     mov    %fs:0x28,%rax
   0x0000000000401076 <+13>:    mov    %rax,0x8(%rsp)
   0x000000000040107b <+18>:    xor    %eax,%eax
   0x000000000040107d <+20>:    lea    0x4(%rsp),%rcx
   0x0000000000401082 <+25>:    mov    %rsp,%rdx
   0x0000000000401085 <+28>:    mov    $0x4025af,%esi
   0x000000000040108a <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x000000000040108f <+38>:    cmp    $0x1,%eax
   0x0000000000401092 <+41>:    jg     0x401099 <phase_5+48>
   0x0000000000401094 <+43>:    callq  0x401426 <explode_bomb>
   0x0000000000401099 <+48>:    mov    (%rsp),%eax
   0x000000000040109c <+51>:    and    $0xf,%eax
   0x000000000040109f <+54>:    mov    %eax,(%rsp)
   0x00000000004010a2 <+57>:    cmp    $0xf,%eax
   0x00000000004010a5 <+60>:    je     0x4010d6 <phase_5+109>
   0x00000000004010a7 <+62>:    mov    $0x0,%ecx
   0x00000000004010ac <+67>:    mov    $0x0,%edx
   0x00000000004010b1 <+72>:    add    $0x1,%edx
   0x00000000004010b4 <+75>:    cltq
   0x00000000004010b6 <+77>:    mov    0x402460(,%rax,4),%eax
   0x00000000004010bd <+84>:    add    %eax,%ecx
   0x00000000004010bf <+86>:    cmp    $0xf,%eax
   0x00000000004010c2 <+89>:    jne    0x4010b1 <phase_5+72>
   0x00000000004010c4 <+91>:    movl   $0xf,(%rsp)
   0x00000000004010cb <+98>:    cmp    $0xf,%edx
   0x00000000004010ce <+101>:   jne    0x4010d6 <phase_5+109>
   0x00000000004010d0 <+103>:   cmp    0x4(%rsp),%ecx
   0x00000000004010d4 <+107>:   je     0x4010db <phase_5+114>
   0x00000000004010d6 <+109>:   callq  0x401426 <explode_bomb>
   0x00000000004010db <+114>:   mov    0x8(%rsp),%rax
   0x00000000004010e0 <+119>:   xor    %fs:0x28,%rax
   0x00000000004010e9 <+128>:   je     0x4010f0 <phase_5+135>
   0x00000000004010eb <+130>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x00000000004010f0 <+135>:   add    $0x18,%rsp
   0x00000000004010f4 <+139>:   retq
End of assembler dump.
(gdb)
```

Process for phase 5:

1.This phase also consists two integers (as we checked in phase 3 and 4)

2.This is like a switch function

3. First make break point at phase_5 and run next each instruction one by one by using (nexti) instruction

4. Analyzing each and every line one by one

and seeing each register values by using info r

5.By considering the first value as 5 we can get the second value as 115

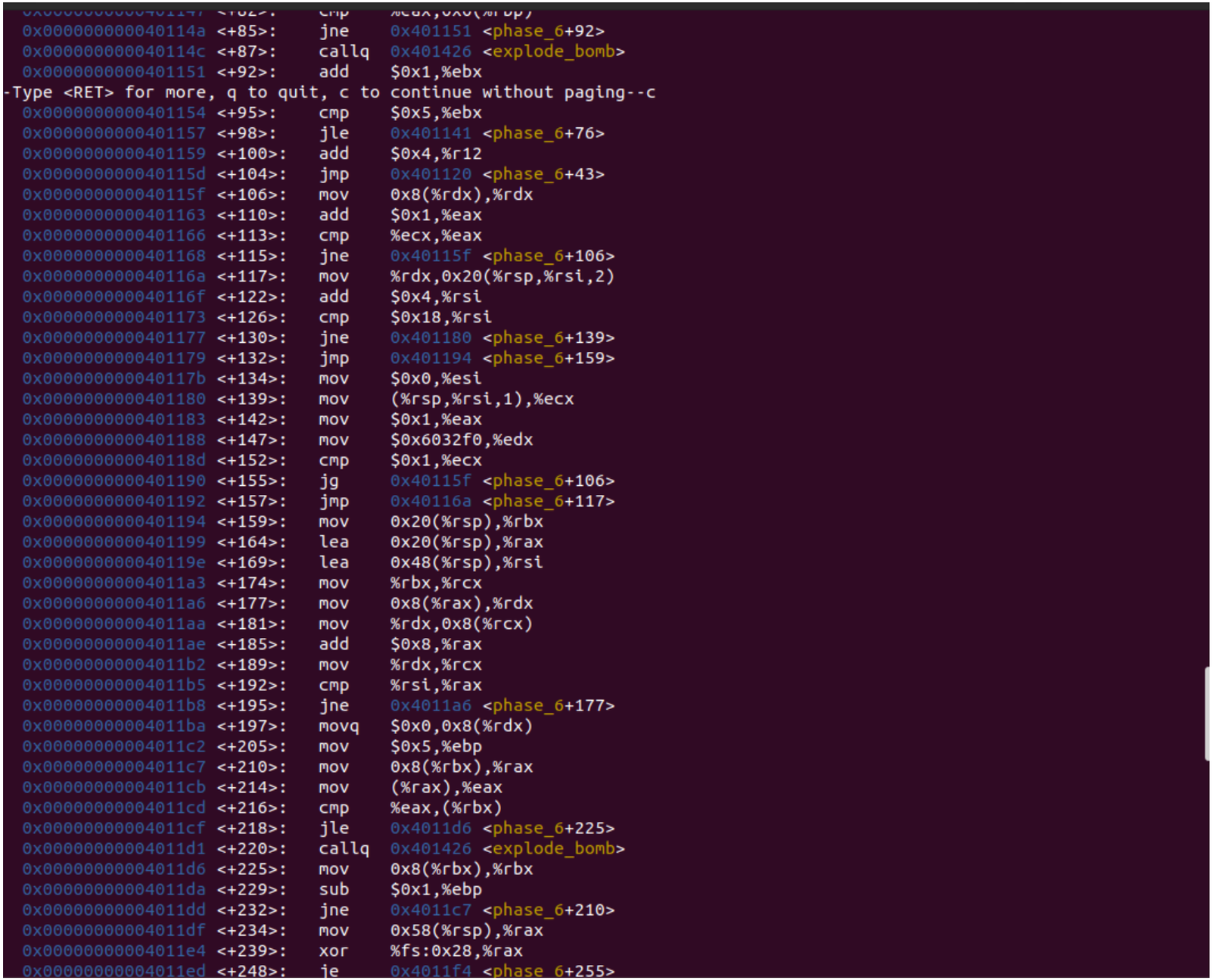
The final answer for phase 5:

5 115

Phase 6:

```
(gdb) b phase_6
Breakpoint 1 at 0x4010f5
(gdb) run answers.txt
Starting program: /home/rahul/Documents/lab_bomb/bomb answers.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
So you got that one.  Try this one.
Good work!  On to the next...
1 5 3 6 2 4

Breakpoint 1, 0x00000000004010f5 in phase_6 ()
(gdb) disas
Dump of assembler code for function phase_6:
=> 0x00000000004010f5 <+0>:     push   %r13
   0x00000000004010f7 <+2>:     push   %r12
   0x00000000004010f9 <+4>:     push   %rbp
   0x00000000004010fa <+5>:     push   %rbx
   0x00000000004010fb <+6>:     sub    $0x68,%rsp
   0x00000000004010ff <+10>:    mov    %fs:0x28,%rax
   0x0000000000401108 <+19>:    mov    %rax,0x58(%rsp)
   0x000000000040110d <+24>:    xor    %eax,%eax
   0x000000000040110f <+26>:    mov    %rsp,%rsi
   0x0000000000401112 <+29>:    callq  0x401448 <read_six_numbers>
   0x0000000000401117 <+34>:    mov    %rsp,%r12
   0x000000000040111a <+37>:    mov    $0x0,%r13d
   0x0000000000401120 <+43>:    mov    %r12,%rbp
   0x0000000000401123 <+46>:    mov    (%r12),%eax
   0x0000000000401127 <+50>:    sub    $0x1,%eax
   0x000000000040112a <+53>:    cmp    $0x5,%eax
   0x000000000040112d <+56>:    jbe    0x401134 <phase_6+63>
   0x000000000040112f <+58>:    callq  0x401426 <explode_bomb>
   0x0000000000401134 <+63>:    add    $0x1,%r13d
   0x0000000000401138 <+67>:    cmp    $0x6,%r13d
   0x000000000040113c <+71>:    je     0x40117b <phase_6+134>
   0x000000000040113e <+73>:    mov    %r13d,%ebx
   0x0000000000401141 <+76>:    movslq %ebx,%rax
   0x0000000000401144 <+79>:    mov    (%rsp,%rax,4),%eax
   0x0000000000401147 <+82>:    cmp    %eax,0x0(%rbp)
   0x000000000040114a <+85>:    jne    0x401151 <phase_6+92>
   0x000000000040114c <+87>:    callq  0x401426 <explode_bomb>
   0x0000000000401151 <+92>:    add    $0x1,%ebx
--Type <RET> for more, q to quit, c to continue without paging--c
```


0x000000000040114a <+85>: jne 0x401151 <phase_6+92>
0x000000000040114c <+87>: callq 0x401426 <explode_bomb>
0x0000000000401151 <+92>: add $0x1,%ebx
-Type <RET> for more, q to quit, c to continue without paging--c
0x0000000000401154 <+95>: cmp $0x5,%ebx
0x0000000000401157 <+98>: jle 0x401141 <phase_6+76>
0x0000000000401159 <+100>: add $0x4,%r12
0x000000000040115d <+104>: jmp 0x401120 <phase_6+43>
0x000000000040115f <+106>: mov 0x8(%rdx),%rdx
0x0000000000401163 <+110>: add $0x1,%eax
0x0000000000401166 <+113>: cmp %ecx,%eax
0x0000000000401168 <+115>: jne 0x40115f <phase_6+106>
0x000000000040116a <+117>: mov %rdx,0x20(%rsp,%rsi,2)
0x000000000040116f <+122>: add $0x4,%rsi
0x0000000000401173 <+126>: cmp $0x18,%rsi
0x0000000000401177 <+130>: jne 0x401180 <phase_6+139>
0x0000000000401179 <+132>: jmp 0x401194 <phase_6+159>
0x000000000040117b <+134>: mov $0x0,%esi
0x0000000000401180 <+139>: mov (%rsp,%rsi,1),%ecx
0x0000000000401183 <+142>: mov $0x1,%eax
0x0000000000401188 <+147>: mov $0x6032f0,%edx
0x000000000040118d <+152>: cmp $0x1,%ecx
0x0000000000401190 <+155>: jg 0x40115f <phase_6+106>
0x0000000000401192 <+157>: jmp 0x40116a <phase_6+117>
0x0000000000401194 <+159>: mov 0x20(%rsp),%rbx
0x0000000000401199 <+164>: lea 0x20(%rsp),%rax
0x000000000040119e <+169>: lea 0x48(%rsp),%rsi
0x00000000004011a3 <+174>: mov %rbx,%rcx
0x00000000004011a6 <+177>: mov 0x8(%rax),%rdx
0x00000000004011aa <+181>: mov %rdx,0x8(%rcx)
0x00000000004011ae <+185>: add $0x8,%rax
0x00000000004011b2 <+189>: mov %rdx,%rcx
0x00000000004011b5 <+192>: cmp %rsi,%rax
0x00000000004011b8 <+195>: jne 0x4011a6 <phase_6+177>
0x00000000004011ba <+197>: movq $0x0,0x8(%rdx)
0x00000000004011c2 <+205>: mov $0x5,%ebp
0x00000000004011c7 <+210>: mov 0x8(%rbx),%rax
0x00000000004011cb <+214>: mov (%rax),%eax
0x00000000004011cd <+216>: cmp %eax,(%rbx)
0x00000000004011cf <+218>: jle 0x4011d6 <phase_6+225>
0x00000000004011d1 <+220>: callq 0x401426 <explode_bomb>
0x00000000004011d6 <+225>: mov 0x8(%rbx),%rbx
0x00000000004011da <+229>: sub $0x1,%ebp
0x00000000004011dd <+232>: jne 0x4011c7 <phase_6+210>
0x00000000004011df <+234>: mov 0x58(%rsp),%rax
0x00000000004011e4 <+239>: xor %fs:0x28,%rax
0x00000000004011ed <+248>: je 0x4011f4 <phase_6+255>

```
   0x00000000004011d6 <+225>:   mov    0x8(%rbx),%rbx
   0x00000000004011da <+229>:   sub    $0x1,%ebp
   0x00000000004011dd <+232>:   jne    0x4011c7 <phase_6+210>
   0x00000000004011df <+234>:   mov    0x58(%rsp),%rax
   0x00000000004011e4 <+239>:   xor    %fs:0x28,%rax
   0x00000000004011ed <+248>:   je     0x4011f4 <phase_6+255>
   0x00000000004011ef <+250>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x00000000004011f4 <+255>:   add    $0x68,%rsp
   0x00000000004011f8 <+259>:   pop    %rbx
   0x00000000004011f9 <+260>:   pop    %rbp
   0x00000000004011fa <+261>:   pop    %r12
   0x00000000004011fc <+263>:   pop    %r13
   0x00000000004011fe <+265>:   retq
End of assembler dump.
(gdb) b *0x00000000004011cd
Breakpoint 2 at 0x4011cd
(gdb) cont
Continuing.

Breakpoint 2, 0x00000000004011cd in phase_6 ()
(gdb) x/3x $esi
0xffffffffffffdf38:     Cannot access memory at address 0xffffffffffffdf38
(gdb) i r
rax            0x19d               413
rbx            0x6032f0            6304496
rcx            0x603320            6304544
rdx            0x603320            6304544
rsi            0x7fffffffdf38      140737488346936
rdi            0x7fffffffd880      140737488345216
rbp            0x5                 0x5
rsp            0x7fffffffdef0      0x7fffffffdef0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f5fac0      140737353480896
r11            0x0                 0
r12            0x7fffffffdf04      140737488346884
r13            0x6                 6
r14            0x0                 0
r15            0x0                 0
rip            0x4011cd            0x4011cd <phase_6+216>
eflags         0x246               [ PF ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x/3x esi
```

```
(gdb) x/3x $rbx
0x6032f0 <node1>:       0x00000379      0x00000001      0x00603330
(gdb) x/3x *($rbx + 8)
0x603330 <node5>:       0x0000019d      0x00000005      0x00603310
(gdb) x/3x *(*($rbx + 8) + 8)
0x603310 <node3>:       0x0000010b      0x00000003      0x00603340
(gdb) x/3x *(*(*($rbx + 8) + 8) +8)
0x603340 <node6>:       0x000000a5      0x00000006      0x00603300
(gdb) x/3x *(*(*(*($rbx + 8) + 8) +8) +8)
0x603300 <node2>:       0x000002cb      0x00000002      0x00603320
(gdb) x/3x *(*(*(*(*($rbx + 8) + 8) +8) +8) +8)
0x603320 <node4>:       0x000003c9      0x00000004      0x00000000
```

Process for phase 6:

1. Set first break point and by using (disas) analyze the code

And in this phase consists 6 numbers and not repeating.

```
0x0000000000401112 <+29>:    callq  0x401448 <read_six_numbers>
0x0000000000401117 <+34>:    mov    %rsp,%r12
0x000000000040111a <+37>:    mov    $0x0,%r13d
0x0000000000401120 <+43>:    mov    %r12,%rbp
0x0000000000401123 <+46>:    mov    (%r12),%eax
0x0000000000401127 <+50>:    sub    $0x1,%eax
0x000000000040112a <+53>:    cmp    $0x5,%eax
```

6 numbers

```
0x000000000040114a <+85>:    jne    0x401151 <phase_6+92>
0x000000000040114c <+87>:    callq  0x401426 <explode_bomb>
0x0000000000401151 <+92>:    add    $0x1,%ebx
Type <RET> for more, q to quit, c to continue without paging--c
0x0000000000401154 <+95>:    cmp    $0x5,%ebx
0x0000000000401157 <+98>:    jle    0x401141 <phase_6+76>
0x0000000000401159 <+100>:   add    $0x4,%r12
0x000000000040115d <+104>:   jmp    0x401120 <phase_6+43>
```

Not repeating

2.By considering the break point at (4001cd)

```
0x00000000004011b8 <+195>:   jne    0x4011a6 <phase_6+177>
0x00000000004011ba <+197>:   movq   $0x0,0x8(%rdx)
0x00000000004011c2 <+205>:   mov    $0x5,%ebp
0x00000000004011c7 <+210>:   mov    0x8(%rbx),%rax
0x00000000004011cb <+214>:   mov    (%rax),%eax
0x00000000004011cd <+216>:   cmp    %eax,(%rbx)
0x00000000004011cf <+218>:   jle    0x4011d6 <phase_6+225>
```

```
End of assembler dump.
(gdb) b *0x00000000004011cd
Breakpoint 2 at 0x4011cd
```

3.By finding values at each memory

```
(gdb) x/3x $rbx
0x6032f0 <node1>:       0x00000379      0x00000001      0x00603330
(gdb) x/3x *($rbx + 8)
0x603330 <node5>:       0x0000019d      0x00000005      0x00603310
(gdb) x/3x *(*($rbx + 8) + 8)
0x603310 <node3>:       0x0000010b      0x00000003      0x00603340
(gdb) x/3x *(*(*($rbx + 8) + 8) +8)
0x603340 <node6>:       0x000000a5      0x00000006      0x00603300
(gdb) x/3x *(*(*(*($rbx + 8) + 8) +8) +8)
0x603300 <node2>:       0x000002cb      0x00000002      0x00603320
(gdb) x/3x *(*(*(*(*($rbx + 8) + 8) +8) +8) +8)
0x603320 <node4>:       0x000003c9      0x00000004      0x00000000
```

And converting these left most values to binary values we get

Node1:889

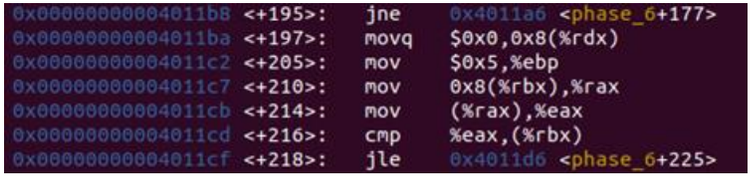
Node5:413

Node3:267

Node6:165

Node2:715

Node4:969



According to the above picture (jle) in the last line

We should keep order of the nodes from smallest to highest

Nodes values (6 <3 <5 <2 <1 <4)

So the required value is 6 3 5 2 1 4

The final answer for phase 6 is:

6 3 5 2 1 4

The final input for 6 phases are :

Phase 1: Houses will begat jobs, jobs will begat houses.

Phase 2: 1 2 4 7 11 16

Phase 3: 4 426

Phase 4: 4 2

Phase 5: 5 115

Phase 6: 6 3 5 2 1 4

THANK YOU