

Name :-

Purval Madhukar Bhude

Roll No. S20230010193

Subject :- CA

Bomb Lab

Bomb No. 304

# Phase 1:

```
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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 phase_1
Breakpoint 1 at 0x15a7
(gdb) run
Starting program: /mnt/c/IIITS ASSIGNMENTS/Sem 2/Computer Arch/Bomb Lab 3/bomb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
hgfd

Breakpoint 1, 0x00005555555555a7 in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x00005555555555a7 <+0>:      endbr64
    0x00005555555555ab <+4>:      sub     $0x8,%rsp
    0x00005555555555af <+8>:      lea     0x1b9a(%rip),%rsi      # 0x555555557150
    0x00005555555555b6 <+15>:     call    0x5555555555aa1 <strings_not_equal>
    0x00005555555555bb <+20>:     test    %eax,%eax
    0x00005555555555bd <+22>:     jne     0x55555555555c4 <phase_1+29>
    0x00005555555555bf <+24>:     add     $0x8,%rsp
    0x00005555555555c3 <+28>:     ret
    0x00005555555555c4 <+29>:     call    0x5555555555bb5 <explode_bomb>
    0x00005555555555c9 <+34>:     jmp     0x55555555555bf <phase_1+24>
End of assembler dump.
(gdb) x/s 0x555555557150
0x555555557150: "Brownie, you are doing a heck of a job."
```

First, set a breakpoint for phase\_1 and run the program. Then, disassemble it, proceeding step by step. In the second line, we check whether the input string is the same as 0x555555557150 and then convert it to a string using x/s

Phase\_1 answer : Brownie, you are doing a heck of a job.

## Phase 2:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) break phase_2
Breakpoint 1 at 0x15cb
(gdb) run
Starting program: /mnt/c/IIITS ASSIGNMENTS/Sem 2/Computer Arch/Bomb Lab 3/bomb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Brownie, you are doing a heck of a job.
Phase 1 defused. How about the next one?
1 223 12
```

```
Breakpoint 1, 0x00005555555555cb in phase_2 ()
(gdb) disas
Dump of assembler code for function phase_2:
=> 0x00005555555555cb <+0>:      endbr64
    0x00005555555555cf <+4>:      push    %rbp
    0x00005555555555d0 <+5>:      push    %rbx
    0x00005555555555d1 <+6>:      sub     $0x28,%rsp
    0x00005555555555d5 <+10>:     mov     %fs:0x28,%rax
    0x00005555555555de <+19>:     mov     %rax,0x18(%rsp)
    0x00005555555555e3 <+24>:     xor     %eax,%eax
    0x00005555555555e5 <+26>:     mov     %rsp,%rsi
    0x00005555555555e8 <+29>:     call   0x5555555555be1 <read_six_numbers>
    0x00005555555555ed <+34>:     cmpl   $0x0,(%rsp)
    0x00005555555555f1 <+38>:     js     0x5555555555fd <phase_2+50>
    0x00005555555555f3 <+40>:     mov     %rsp,%rbp
    0x00005555555555f6 <+43>:     mov     $0x1,%ebx
    0x00005555555555fb <+48>:     jmp     0x5555555555615 <phase_2+74>
    0x00005555555555fd <+50>:     call   0x5555555555bb5 <explode_bomb>
    0x0000555555555602 <+55>:     jmp     0x5555555555f3 <phase_2+40>
    0x0000555555555604 <+57>:     call   0x5555555555bb5 <explode_bomb>
    0x0000555555555609 <+62>:     add     $0x1,%ebx
    0x000055555555560c <+65>:     add     $0x4,%rbp
    0x0000555555555610 <+69>:     cmp     $0x6,%ebx
    0x0000555555555613 <+72>:     je      0x5555555555621 <phase_2+86>
    0x0000555555555615 <+74>:     mov     %ebx,%eax
    0x0000555555555617 <+76>:     add     0x0(%rbp),%eax
    0x000055555555561a <+79>:     cmp     %eax,0x4(%rbp)
    0x000055555555561d <+82>:     je      0x5555555555609 <phase_2+62>
    0x000055555555561f <+84>:     jmp     0x5555555555604 <phase_2+57>
    0x0000555555555621 <+86>:     mov     0x18(%rsp),%rax
    0x0000555555555626 <+91>:     xor     %fs:0x28,%rax
    0x000055555555562f <+100>:    jne     0x5555555555638 <phase_2+109>
    0x0000555555555631 <+102>:    add     $0x28,%rsp
    0x0000555555555635 <+106>:    pop     %rbx
    0x0000555555555636 <+107>:    pop     %rbp
    0x0000555555555637 <+108>:    ret
    0x0000555555555638 <+109>:    call   0x5555555555220 <__stack_chk_fail@plt>
End of assembler dump.
(gdb) stepi
0x00005555555555cf in phase_2 ()
(gdb) nexti
```

Set a breakpoint for phase\_2 and then run the program. Upon disassembling it, we receive a hint indicating that there are six numbers expected in the answer, inferred from the function name (read\_six\_numbers). After stepping through the code and disassembling it seven times, we enter the read\_six\_numbers function.

```

End of assembler dump.
(gdb) print/d $eax
$1 = 1
(gdb) nexti
0x000055555555561a in phase_2 ()
(gdb) print/d $eax
$2 = 2
(gdb) x/d $rbp+4
0x7fffffffef004: 2
(gdb) nexti
0x000055555555561d in phase_2 ()
(gdb) nexti
0x0000555555555609 in phase_2 ()
(gdb) nexti
0x000055555555560c in phase_2 ()
(gdb) nexti
0x0000555555555610 in phase_2 ()
(gdb) nexti
0x0000555555555613 in phase_2 ()
(gdb) nexti
0x0000555555555615 in phase_2 ()
(gdb) nexti
0x0000555555555617 in phase_2 ()
(gdb) print/d $eax
$3 = 2
(gdb) nexti
0x000055555555561a in phase_2 ()
(gdb) x/d $rbp+4
0x7fffffffef008: 3
(gdb) nexti
A syntax error in expression, near `%eax'.
(gdb) print/d $eax
$1 = 4
(gdb) nexti
0x0000555555555617 in phase_2 ()
(gdb) nexti
0x000055555555561a in phase_2 ()
(gdb) nexti
0x000055555555561d in phase_2 ()
(gdb) nexti
0x0000555555555609 in phase_2 ()
(gdb) nexti
0x000055555555560c in phase_2 ()
(gdb) nexti
0x0000555555555610 in phase_2 ()
(gdb) nexti
0x0000555555555613 in phase_2 ()
(gdb) nexti
0x0000555555555615 in phase_2 ()
(gdb) print/d $eax
$2 = 7
(gdb) nexti
0x0000555555555617 in phase_2 ()
(gdb) nexti
0x000055555555561a in phase_2 ()
(gdb) nexti
0x000055555555561d in phase_2 ()
(gdb) nexti
0x0000555555555609 in phase_2 ()
(gdb) nexti
0x000055555555560c in phase_2 ()
(gdb) nexti
0x0000555555555610 in phase_2 ()
(gdb) nexti
0x0000555555555613 in phase_2 ()
(gdb) nexti
0x0000555555555615 in phase_2 ()
(gdb) print/d $eax
$3 = 11

```

And then analysing the function we understand that it is  $eax = eax + index$  which can also be written as  $eax += index$  then find 6 numbers

$eax = 1 + 0 = 1$  (index = 0)

$eax = 1 + 1 = 2$  (index = 1)

$eax = 2 + 2 = 4$  (index = 2)

$eax = 4 + 3 = 7$  (index = 3)

$eax = 7 + 4 = 11$  (index = 4)

$eax = 11 + 5 = 16$  (index = 5)

Phase 2 answer: 1 2 4 7 11 16

## Phase 3:

```
root@kali:~/Desktop# gdb ./bomblab
GNU gdb (Ubuntu 12.1-ubuntu12.04) 12.1
Copyright (C) 2022 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:
<https://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 bomblab...
(gdb) break phase_3
Breakpoint 1 at 0x555555563d
(gdb) run
Starting program: /mnt/c/IIITS ASSIGNMENTS/sem 2/Computer Arch/Bomb Lab 3/bomblab
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Brownie, you are doing a heck of a job.
Phase 1 defused. How about the next one?
1 2 4 7 11 16
That's number 2. Keep going!
5 n 269

Breakpoint 1, 0x000055555555563d in phase_3 ()
(gdb) disas
Dump of assembler code for function phase_3:
--
0x0000555555555641 <+41>:    andb $0, %rax
0x0000555555555645 <+45>:    mov     %rsi, %rax
0x0000555555555649 <+49>:    mov     %rax, %eax
0x0000555555555653 <+53>:    xor     %eax, %eax
0x0000555555555657 <+57>:    lea     0x4(%rsp), %rcx
0x000055555555565b <+5b>:    mov     %rsp, %rdi
0x000055555555565f <+5f>:    call    0x55555555730f <__isoc99_sscanf@plt>
0x0000555555555663 <+63>:    cmp     $0x1, %eax
0x0000555555555667 <+67>:    jle     0x555555555688 <phase_3+75>
0x000055555555566b <+6b>:    jmp     0x5555555556d9 <phase_3+15d>
0x000055555555566f <+6f>:    lea     0x1b52(%rip), %rdx
0x0000555555555673 <+73>:    movslq (%rdx,%rax,4), %rax
0x0000555555555677 <+77>:    add     %rdx, %rax
0x000055555555567b <+7b>:    notrack jmp     %rax
0x000055555555567f <+7f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555683 <+83>:    jmp     0x55555555566e <phase_3+49>
0x0000555555555687 <+87>:    mov     %eax, %eax
0x000055555555568b <+8b>:    cmp     %eax, 0x4(%rsp)
0x000055555555568f <+8f>:    jne     0x5555555556ec <phase_3+175>
0x0000555555555693 <+93>:    mov     0x8(%rsp), %rax
0x0000555555555697 <+97>:    or      %rsi, %rax
0x000055555555569b <+9b>:    jne     0x5555555556f3 <phase_3+182>
0x000055555555569f <+9f>:    add     $0x18, %rsp
0x00005555555556a3 <+a3>:    ret
0x00005555555556a7 <+a7>:    mov     $0xd5, %eax
0x00005555555556ab <+ab>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556af <+af>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556b3 <+b3>:    mov     $0x302, %eax
0x00005555555556b7 <+b7>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556bb <+bb>:    mov     $0x278, %eax
0x00005555555556bf <+bf>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556c3 <+c3>:    mov     $0x2e6, %eax
0x00005555555556c7 <+c7>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556cb <+cb>:    mov     $0x342, %eax
0x00005555555556cf <+cf>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556d3 <+d3>:    jmp     0x55555555566e <phase_3+49>
0x00005555555556d7 <+d7>:    call    0x5555555556b5 <explode_bomb>
0x00005555555556db <+db>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556df <+df>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556e3 <+e3>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556e7 <+e7>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556eb <+eb>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556ef <+ef>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556f3 <+f3>:    call    0x5555555556b5 <explode_bomb>
0x00005555555556f7 <+f7>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556fb <+fb>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556ff <+ff>:    jmp     0x555555555694 <phase_3+87>
--
Type "next" for more.
Type "quit" to continue without paging.-c
0x00005555555556a3 <+a3>:    mov     $0x12f, %eax
0x00005555555556a7 <+a7>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556ab <+ab>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556af <+af>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556b3 <+b3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556b7 <+b7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556bb <+bb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556bf <+bf>:    call    0x5555555556b5 <explode_bomb>
0x00005555555556c3 <+c3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556c7 <+c7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556cb <+cb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556cf <+cf>:    call    0x5555555556b5 <explode_bomb>
0x00005555555556d3 <+d3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556d7 <+d7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556db <+db>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556df <+df>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556e3 <+e3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556e7 <+e7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556eb <+eb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556ef <+ef>:    call    0x5555555556b5 <explode_bomb>
0x00005555555556f3 <+f3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556f7 <+f7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555556fb <+fb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555556ff <+ff>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555703 <+03>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555707 <+07>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555570b <+0b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555570f <+0f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555713 <+13>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555717 <+17>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555571b <+1b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555571f <+1f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555723 <+23>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555727 <+27>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555572b <+2b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555572f <+2f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555733 <+33>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555737 <+37>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555573b <+3b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555573f <+3f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555743 <+43>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555747 <+47>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555574b <+4b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555574f <+4f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555753 <+53>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555757 <+57>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555575b <+5b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555575f <+5f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555763 <+63>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555767 <+67>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555576b <+6b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555576f <+6f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555773 <+73>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555777 <+77>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555577b <+7b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555577f <+7f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555783 <+83>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555787 <+87>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555578b <+8b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555578f <+8f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555793 <+93>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555797 <+97>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555579b <+9b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555579f <+9f>:    call    0x5555555556b5 <explode_bomb>
0x00005555555557a3 <+a3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557a7 <+a7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555557ab <+ab>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557af <+af>:    call    0x5555555556b5 <explode_bomb>
0x00005555555557b3 <+b3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557b7 <+b7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555557bb <+bb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557bf <+bf>:    call    0x5555555556b5 <explode_bomb>
0x00005555555557c3 <+c3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557c7 <+c7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555557cb <+cb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557cf <+cf>:    call    0x5555555556b5 <explode_bomb>
0x00005555555557d3 <+d3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557d7 <+d7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555557db <+db>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557df <+df>:    call    0x5555555556b5 <explode_bomb>
0x00005555555557e3 <+e3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557e7 <+e7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555557eb <+eb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557ef <+ef>:    call    0x5555555556b5 <explode_bomb>
0x00005555555557f3 <+f3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557f7 <+f7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555557fb <+fb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555557ff <+ff>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555803 <+03>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555807 <+07>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555580b <+0b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555580f <+0f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555813 <+13>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555817 <+17>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555581b <+1b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555581f <+1f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555823 <+23>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555827 <+27>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555582b <+2b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555582f <+2f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555833 <+33>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555837 <+37>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555583b <+3b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555583f <+3f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555843 <+43>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555847 <+47>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555584b <+4b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555584f <+4f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555853 <+53>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555857 <+57>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555585b <+5b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555585f <+5f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555863 <+63>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555867 <+67>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555586b <+6b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555586f <+6f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555873 <+73>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555877 <+77>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555587b <+7b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555587f <+7f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555883 <+83>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555887 <+87>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555588b <+8b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555588f <+8f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555893 <+93>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555897 <+97>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555589b <+9b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555589f <+9f>:    call    0x5555555556b5 <explode_bomb>
0x00005555555558a3 <+a3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558a7 <+a7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555558ab <+ab>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558af <+af>:    call    0x5555555556b5 <explode_bomb>
0x00005555555558b3 <+b3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558b7 <+b7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555558bb <+bb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558bf <+bf>:    call    0x5555555556b5 <explode_bomb>
0x00005555555558c3 <+c3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558c7 <+c7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555558cb <+cb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558cf <+cf>:    call    0x5555555556b5 <explode_bomb>
0x00005555555558d3 <+d3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558d7 <+d7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555558db <+db>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558df <+df>:    call    0x5555555556b5 <explode_bomb>
0x00005555555558e3 <+e3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558e7 <+e7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555558eb <+eb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558ef <+ef>:    call    0x5555555556b5 <explode_bomb>
0x00005555555558f3 <+f3>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558f7 <+f7>:    jmp     0x5555555556b5 <explode_bomb>
0x00005555555558fb <+fb>:    jmp     0x555555555694 <phase_3+87>
0x00005555555558ff <+ff>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555903 <+03>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555907 <+07>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555590b <+0b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555590f <+0f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555913 <+13>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555917 <+17>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555591b <+1b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555591f <+1f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555923 <+23>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555927 <+27>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555592b <+2b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555592f <+2f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555933 <+33>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555937 <+37>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555593b <+3b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555593f <+3f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555943 <+43>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555947 <+47>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555594b <+4b>:    jmp     0x555555555694 <phase_3+87>
0x000055555555594f <+4f>:    call    0x5555555556b5 <explode_bomb>
0x0000555555555953 <+53>:    jmp     0x555555555694 <phase_3+87>
0x0000555555555957 <+57>:    jmp     0x5555555556b5 <explode_bomb>
0x000055555555595b <+5b>:    jmp     0x5555555
```

```

0x0000555555555682 <+69>: add    %rdx,%rax
0x0000555555555685 <+72>: notrack jmp *%rax
0x0000555555555688 <+75>: call   0x555555555bb5 <explode_bomb>
0x000055555555568d <+80>: jmp     0x5555555556e <phase_3+49>
0x000055555555568f <+82>: mov     $0xd2,%eax
0x0000555555555694 <+87>: cmp     %eax,0x4(%rsp)
0x0000555555555698 <+91>: jne     0x5555555556ec <phase_3+175>
0x000055555555569a <+93>: mov     0x8(%rsp),%rax
0x000055555555569f <+98>: xor     %fs:0x28,%rax
0x00005555555556a8 <+107>: jne     0x5555555556f3 <phase_3+182>
0x00005555555556aa <+109>: add     $0x18,%rsp
0x00005555555556ae <+113>: ret
=> 0x00005555555556af <+114>: mov     $0x1db,%eax
0x00005555555556b4 <+119>: jmp     0x555555555694 <phase_3+87>
0x00005555555556b6 <+121>: mov     $0xd5,%eax
0x00005555555556bb <+126>: jmp     0x555555555694 <phase_3+87>

```

(gdb) print/d 0x1db  
\$1 = 475

For first number when it is 2 answer is  
Phase 2: 2 475

## Phase 4:

```

[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Brownie, you are doing a heck of a job.
1 2 4 7Phase 1 defused. How about the next one?
11 16
1That's number 2. Keep going!
210
Halfway there!
12 12

Breakpoint 1, 0x000055555555572e in phase_4 ()
(gdb) disas
Dump of assembler code for function phase_4:
=> 0x000055555555572e <+0>: endbr64
0x0000555555555732 <+4>: sub     $0x18,%rsp
0x0000555555555736 <+8>: mov     %fs:0x28,%rax
0x000055555555573f <+17>: mov     %rax,0x8(%rsp)
0x0000555555555744 <+22>: xor     %eax,%eax
0x0000555555555746 <+24>: lea     0x4(%rsp),%rcx
0x000055555555574b <+29>: mov     %rsp,%rdx
0x000055555555574e <+32>: lea     0x1bba(%rip),%rsi # 0x555555555730f
0x0000555555555755 <+39>: call    0x5555555552c0 <__isoc99_sscanf@plt>
0x000055555555575a <+44>: cmp     $0x2,%eax
0x000055555555575d <+47>: jne     0x555555555765 <phase_4+55>
0x000055555555575f <+49>: cmpl    $0xe,(%rsp)
0x0000555555555763 <+53>: jbe     0x55555555576a <phase_4+60>
0x0000555555555765 <+55>: call    0x555555555bb5 <explode_bomb>
0x000055555555576a <+60>: mov     $0xe,%edx
0x000055555555576f <+65>: mov     $0x0,%esi
0x0000555555555774 <+70>: mov     (%rsp),%edi
0x0000555555555777 <+73>: call    0x5555555556f8 <func4>
0x000055555555577c <+78>: cmp     $0x12,%eax
0x000055555555577f <+81>: jne     0x555555555788 <phase_4+90>
0x0000555555555781 <+83>: cmpl    $0x12,0x4(%rsp)
0x0000555555555786 <+88>: je      0x55555555578d <phase_4+95>
0x0000555555555788 <+90>: call    0x555555555bb5 <explode_bomb>
0x000055555555578d <+95>: mov     0x8(%rsp),%rax
0x0000555555555792 <+100>: xor     %fs:0x28,%rax
0x000055555555579b <+109>: jne     0x5555555557a2 <phase_4+116>
0x000055555555579d <+111>: add     $0x18,%rsp
0x00005555555557a1 <+115>: ret
0x00005555555557a2 <+116>: call    0x555555555220 <__stack_chk_fail@plt>
End of assembler dump.

```



Similarway, putting break point and running then disas. Here we known that

```
(gdb) x/s 0x55555555730f
0x55555555730f: "%d %d"
```

there are 2 integer as a input

```
0x00005555555574e <+32>:    lea    0x1bba(%rip),%rsi    # 0x55555555730f
0x000055555555755 <+39>:    call   0x5555555552c0 <__isoc99_sscanf@plt>
0x00005555555575a <+44>:    cmp    $0x2,%eax
0x00005555555575d <+47>:    jne    0x55555555765 <phase_4+55>
0x00005555555575f <+49>:    cmpl   $0xe,(%rsp)
=> 0x000055555555763 <+53>:    jbe    0x5555555576a <phase_4+60>
0x000055555555765 <+55>:    call   0x555555555bb5 <explode_bomb>
0x00005555555576a <+60>:    mov    $0xe,%edx
0x00005555555576f <+65>:    mov    $0x0,%esi
0x000055555555774 <+70>:    mov    (%rsp),%edi
0x000055555555777 <+73>:    call   0x5555555556f8 <func4>
```

By this part we understand that first number will be in range of 2 -> 14 (e) and by going in recursive function func4 for 11 as a input we it is giving output as 0x12 for which it is check you can see that below in the image as well so we by hit and trial between 2->14 we can say that first number is 11.

```
0x00005555555576f <+65>:    mov    $0x0,%esi
0x000055555555774 <+70>:    mov    (%rsp),%edi
0x000055555555777 <+73>:    call   0x5555555556f8 <func4>
0x00005555555577c <+78>:    cmp    $0x12,%eax
=> 0x00005555555577f <+81>:    jne    0x55555555788 <phase_4+90>
0x000055555555781 <+83>:    cmpl   $0x12,0x4(%rsp)
```

Now, to find 2<sup>nd</sup> number we can easily see that it is comparing we 0x12 and check whether they are

equal or not so second number is 0x12 which is 18. 

```
(gdb) print /d 0x12
$1 = 18
```

```
0x000055555555774 <+70>:    mov    (%rsp),%edi
0x000055555555777 <+73>:    call   0x5555555556f8 <func4>
0x00005555555577c <+78>:    cmp    $0x12,%eax
0x00005555555577f <+81>:    jne    0x55555555788 <phase_4+90>
0x000055555555781 <+83>:    cmpl   $0x12,0x4(%rsp)
0x000055555555786 <+88>:    je     0x5555555578d <phase_4+95>
0x000055555555788 <+90>:    call   0x555555555bb5 <explode_bomb>
0x00005555555578d <+95>:    mov    0x8(%rsp),%rax
0x000055555555792 <+100>:   xor    %fs:0x28,%rax
0x000055555555794 <+102>:    jmp     0x5555555555b2 <_start+116>
```

## Phase 5:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) break phase_5
Breakpoint 1 at 0x17a7
(gdb) run
Starting program: /mnt/c/IIITS ASSIGNMENTS/Sem 2/Computer Arch/Bomb Lab 3/bomb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Brownie, you are doing a heck of a job.
1 2 4 7Phase 1 defused. How about the next one?
 11 16
1That's number 2. Keep going!
 210
11 Halfway there!
 18
So you got that one. Try this one.
123 123

Breakpoint 1, 0x0000555555557a7 in phase_5 ()
(gdb) disas
Dump of assembler code for function phase_5:
=> 0x0000555555557a7 <+0>:      endbr64
    0x0000555555557ab <+4>:      push   %rbx
    0x0000555555557ac <+5>:      mov     %rdi,%rbx
    0x0000555555557af <+8>:      call    0x55555555a80 <string_length>
    0x0000555555557b4 <+13>:     cmp     $0x6,%eax
    0x0000555555557b7 <+16>:     jne     0x555555557e5 <phase_5+62>
    0x0000555555557b9 <+18>:     mov     %rbx,%rax
    0x0000555555557bc <+21>:     lea     0x6(%rbx),%rdi
    0x0000555555557c0 <+25>:     mov     $0x0,%ecx
    0x0000555555557c5 <+30>:     lea     0x19f4(%rip),%rsi      # 0x5555555571c0 <array.3471>
    0x0000555555557cc <+37>:     movzbl (%rax),%edx
    0x0000555555557cf <+40>:     and     $0xf,%edx
    0x0000555555557d2 <+43>:     add     (%rsi,%rdx,4),%ecx
    0x0000555555557d5 <+46>:     add     $0x1,%rax
    0x0000555555557d9 <+50>:     cmp     %rdi,%rax
    0x0000555555557dc <+53>:     jne     0x555555557cc <phase_5+37>
    0x0000555555557de <+55>:     cmp     $0x3b,%ecx
    0x0000555555557e1 <+58>:     jne     0x555555557ec <phase_5+69>
    0x0000555555557e3 <+60>:     pop     %rbx
    0x0000555555557e4 <+61>:     ret
    0x0000555555557e5 <+62>:     call    0x55555555bb5 <explode_bomb>
    0x0000555555557ea <+67>:     jmp     0x555555557b9 <phase_5+18>
    0x0000555555557ec <+69>:     call    0x55555555bb5 <explode_bomb>
    0x0000555555557f1 <+74>:     jmp     0x555555557e3 <phase_5+60>
End of assembler dump.
```

We put breakpoint, run and then disas. Here we understand that it is string or array and in next line it is comparing with 0x6 which meaning that array or string must be of length 6.



```

(gdb) x/d 0x5555555571c0
0x5555555571c0 <array.3471>: 2
(gdb)
0x5555555571c1 <array.3471+1>: 0
(gdb) x/d 0x5555555571c0
0x5555555571c0 <array.3471>: 2
(gdb) x/d 0x5555555571c4
0x5555555571c4 <array.3471+4>: 10
(gdb) x/d 0x5555555571c8
0x5555555571c8 <array.3471+8>: 6
(gdb) x/d 0x5555555571cc
0x5555555571cc <array.3471+12>: 1
(gdb) x/d 0x5555555571d0
0x5555555571d0 <array.3471+16>: 12
(gdb) x/d 0x5555555571d4
0x5555555571d4 <array.3471+20>: 16
(gdb) x/d 0x5555555571d8
0x5555555571d8 <array.3471+24>: 9
(gdb) x/d 0x5555555571dc
0x5555555571dc <array.3471+28>: 3
(gdb) x/d 0x5555555571e0
0x5555555571e0 <array.3471+32>: 4
(gdb) x/d 0x5555555571e4
0x5555555571e4 <array.3471+36>: 7
(gdb) print /d 0x3b
$2 = 59

```

By array's first pointer we can find all the array till 10<sup>th</sup> position giving above image. By seeing it we understand that sum of all input of array should be equal to 0x3b. string store index. So we make one of it as 111116 as

sum = arr[1]+ arr[1]+ arr[1]+ arr[1]+ arr[1]+ arr[6]

= 10 + 10 + 10 + 10 + 10 + 9 = 59

We can make any combination but sum should be 59.

Phase 5 answer: 111116

## Phase 6:

```
(gdb) run
Starting program: /mnt/c/IIITS ASSIGNMENTS/Sem 2/Computer Arch/Bomb Lab 3/bomb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Brownie, you are doing a heck of a job.
1 2 4 7 Phase 1 defused. How about the next one?
11 16
1That's number 2. Keep going!
210
11 Halfway there!
18
11111So you got that one. Try this one.
6
Good work! On to the next...
1 2 3 4 5 6

Breakpoint 1, 0x00005555555557f3 in phase_6 ()
(gdb) disas
Dump of assembler code for function phase_6:
=> 0x00005555555557f3 <+0>:    endbr64
0x00005555555557f7 <+4>:    push   %r15
0x00005555555557f9 <+6>:    push   %r14
0x00005555555557fb <+8>:    push   %r13
0x00005555555557fd <+10>:   push   %r12
0x00005555555557ff <+12>:   push   %rbp
0x0000555555555800 <+13>:   push   %rbx
0x0000555555555801 <+14>:   sub    $0x68,%rsp
0x0000555555555805 <+18>:   mov    %fs:0x28,%rax
0x000055555555580a <+22>:   mov    %rax,0x58(%rsp)
0x0000555555555813 <+32>:   xor    %eax,%eax
0x0000555555555815 <+34>:   mov    %rsp,%r14
0x0000555555555818 <+37>:   mov    %r14,%rsi
0x000055555555581b <+40>:   call   0x5555555558b1 <read_six_numbers>
0x0000555555555820 <+45>:   mov    %r14,%r12
0x0000555555555823 <+48>:   mov    %r14,%r1d
0x0000555555555829 <+54>:   mov    %rsp,%r13
0x000055555555582e <+59>:   jmp     0x5555555558f2 <phase_6+255>
0x0000555555555831 <+62>:   call   0x5555555558b5 <explode_bomb>
0x0000555555555836 <+67>:   jmp     0x555555555894 <phase_6+273>
0x000055555555583b <+72>:   call   0x5555555558b5 <explode_bomb>
0x0000555555555840 <+77>:   add    $0x1,%rbx
0x000055555555584a <+81>:   cmp    $0x5,%ebx
0x0000555555555847 <+84>:   jg      0x5555555558ea <phase_6+247>
0x000055555555584a <+90>:   mov    0x0(%r13,%rbx,4),%eax
0x0000555555555852 <+95>:   cmp    %eax,0x0(%rbp)
0x0000555555555855 <+98>:   jne     0x555555555840 <phase_6+77>
0x0000555555555857 <+100>:  jmp     0x55555555583b <phase_6+72>
0x0000555555555859 <+102>:  lea     0x18(%r12),%rcx
0x000055555555585e <+107>:  mov    %edx,%edx
0x0000555555555863 <+112>:  sub     (%r12),%eax
0x0000555555555865 <+114>:  sub     (%r12),%eax
0x0000555555555869 <+118>:  mov    %eax,0x12(%r12)
0x000055555555586d <+122>:  add    $0x4,%r12
0x0000555555555871 <+126>:  cmp    %r12,%rcx
0x000055555555587a <+129>:  jne     0x555555555863 <phase_6+112>
0x000055555555587e <+131>:  mov    $0x0,%esi
0x000055555555587b <+136>:  mov    (%rsp,%rsi,4),%ecx
0x000055555555587e <+139>:  mov    $0x1,%eax
0x0000555555555882 <+144>:  lea     0x3984(%rip),%rdx    # 0x555555559210 <node1>
0x000055555555588a <+151>:  cmp    $0x1,%ecx
0x000055555555588d <+154>:  jle     0x55555555589a <phase_6+167>
0x000055555555588f <+156>:  mov    0x8(%rdx),%rdx
0x0000555555555893 <+160>:  add    $0x1,%eax
0x0000555555555896 <+163>:  cmp    %ecx,%eax
0x0000555555555898 <+165>:  jne     0x55555555588f <phase_6+156>
0x000055555555589a <+167>:  mov    %rdx,0x20(%rsp,%rsi,8)
0x000055555555589f <+172>:  add    $0x1,%rsi
--Type <RET> for more, q to quit, c to continue without paging--c
0x00005555555558a3 <+176>:  cmp    $0x6,%rsi
0x00005555555558a7 <+180>:  jne     0x55555555587b <phase_6+136>
0x00005555555558a9 <+182>:  mov     0x20(%rsp),%rbx
0x00005555555558aa <+187>:  mov     0x28(%rsp),%rax
0x00005555555558ab <+192>:  mov     %rax,0x8(%rbx)
0x00005555555558ab <+196>:  mov     0x30(%rsp),%rdx
0x00005555555558bc <+201>:  mov     %rdx,0x8(%rax)
0x00005555555558c0 <+205>:  mov     0x38(%rsp),%rax
0x00005555555558c5 <+210>:  mov     %rax,0x8(%rdx)
0x00005555555558c9 <+214>:  mov     0x40(%rsp),%rdx
0x00005555555558cc <+219>:  mov     %rdx,0x8(%rax)
0x00005555555558d2 <+223>:  mov     0x48(%rsp),%rax
0x00005555555558d7 <+228>:  mov     %rax,0x8(%rdx)
0x00005555555558db <+232>:  movq    $0x0,0x8(%rax)
0x00005555555558de <+240>:  mov     $0x5,%ebp
0x00005555555558e0 <+245>:  jmp     0x55555555591f <phase_6+308>
0x00005555555558e2 <+247>:  add     $0x1,%r15
0x00005555555558e6 <+251>:  add     $0x4,%r14
0x00005555555558f2 <+255>:  mov     %r14,%rbp
0x00005555555558f5 <+258>:  mov     (%r14),%eax
0x00005555555558f6 <+261>:  sub     $0x1,%eax
0x00005555555558fb <+264>:  cmp     $0x5,%eax
0x00005555555558fe <+267>:  ja      0x555555555831 <phase_6+62>
0x00005555555558ff <+273>:  cmp     $0x5,%r1d
0x0000555555555900 <+277>:  jg      0x555555555859 <phase_6+102>
0x0000555555555906 <+283>:  mov     %r15,%rbx
0x0000555555555911 <+286>:  jmp     0x55555555584d <phase_6+90>
0x0000555555555916 <+291>:  mov     0x8(%rbx),%rbx
0x000055555555591a <+295>:  sub     $0x1,%ebp
0x000055555555591d <+298>:  je      0x555555555930 <phase_6+317>
0x000055555555591f <+300>:  mov     0x8(%rbx),%rax
0x0000555555555923 <+304>:  mov     (%rax),%eax
0x0000555555555925 <+306>:  cmp     %eax,0x8(%rbx)
0x0000555555555927 <+308>:  jge     0x555555555916 <phase_6+291>
0x0000555555555929 <+310>:  call    0x5555555558b5 <explode_bomb>
0x000055555555592a <+315>:  jmp     0x555555555916 <phase_6+291>
0x0000555555555930 <+317>:  mov     0x58(%rsp),%rax
0x0000555555555935 <+322>:  xor     %fs:0x28,%rax
0x0000555555555936 <+331>:  jne     0x55555555594f <phase_6+348>
0x0000555555555940 <+333>:  add     $0x68,%rsp
0x000055555555594a <+337>:  pop     %rbp
0x000055555555594b <+338>:  pop     %rbx
0x000055555555594e <+339>:  pop     %r12
0x000055555555594b <+341>:  pop     %r13
0x000055555555594c <+343>:  pop     %r14
0x000055555555594c <+345>:  pop     %r15
0x000055555555594e <+347>:  ret
0x000055555555594f <+348>:  call    0x555555555820 <__stack_chk_fail@plt>
End of assembler dump.
(gdb) nexti 16
0x0000555555555899 in phase_6 ()
```

Same as previous we have to do breakpoint, run and then disas by analysing the above it is using linked list and if we take out nodes which is shown in below image.

```
(gdb) x/12gx 0x555555559210
0x555555559210 <node1>: 0x00000001000001fe 0x0000555555559220
0x555555559220 <node2>: 0x0000000200000195 0x0000555555559230
0x555555559230 <node3>: 0x000000030000022f 0x0000555555559240
0x555555559240 <node4>: 0x00000004000001b8 0x0000555555559250
0x555555559250 <node5>: 0x000000050000031c 0x0000555555559110
0x555555559260 <host_table>: 0x0000555555557369 0x0000555555557383
(gdb) x/1gx 0x555555559110
0x555555559110 <node6>: 0x0000000600000094
```

```
0x0000555555555925 <+306>:  cmp    %eax,0x8(%rbx)
0x0000555555555927 <+308>:  jge     0x555555555916 <phase_6+291>
0x0000555555555929 <+310>:  call    0x5555555558b5 <explode_bomb>
```

By this image we can say that order should be decreasing order which we found using above image and that numbers are 5 3 1 4 2 6.

```
0x0000555555555859 <+102>:  lea     0x18(%r12),%rcx
0x000055555555585e <+107>:  mov     $0x7,%edx
0x0000555555555863 <+112>:  mov     %edx,%eax
0x0000555555555865 <+114>:  sub     (%r12),%eax
0x0000555555555869 <+118>:  mov     %eax,0x12(%r12)
0x000055555555586d <+122>:  add     $0x4,%r12
0x0000555555555871 <+126>:  cmp     %r12,%rcx
```

And then we also have to do 7- number that will be the answer.

arr[0] = 7-5 = 2

arr[1] = 7-3 = 4

arr[2] = 7-1 = 6

arr[3] = 7-4 = 3

arr[4] = 7-2 = 5

arr[5] = 7-6 = 1

Phase 6 answer: 2 4 6 3 5 1

## Secret Phase:

```
type -apropos word to search for commands
Reading symbols from bomb...
(gdb) break phase_4
Breakpoint 1 at 0x172e
(gdb) breal secret_phase
Undefined command: "breal". Try "help".
(gdb) break secret_phase
Breakpoint 2 at 0x1995
```

Setting the break point in phase\_4 and then in secret\_phase

And then running the program and disas the secret phase

```
Breakpoint 2, 0x0000555555555995 in secret_phase ()
(gdb) disas
Dump of assembler code for function secret_phase:
=> 0x0000555555555995 <+0>:      endbr64
0x0000555555555999 <+4>:      push    %rbx
0x000055555555599a <+5>:      call   0x555555555c26 <read_line>
0x000055555555599f <+10>:     mov     %rax,%rdi
0x00005555555559a2 <+13>:     mov     $0xa,%edx
0x00005555555559a7 <+18>:     mov     $0x0,%esi
0x00005555555559ac <+23>:     call   0x5555555552a0 <strtol@plt>
0x00005555555559b1 <+28>:     mov     %rax,%rbx
0x00005555555559b4 <+31>:     lea     -0x1(%rax),%eax
0x00005555555559b7 <+34>:     cmp     $0x3e8,%eax
0x00005555555559bc <+39>:     ja      0x5555555559e3 <secret_phase+78>
0x00005555555559be <+41>:     mov     %ebx,%esi
0x00005555555559c0 <+43>:     lea     0x3769(%rip),%rdi      # 0x5555555559130 <n1>
0x00005555555559c7 <+50>:     call   0x555555555954 <fun7>
0x00005555555559cc <+55>:     test    %eax,%eax
0x00005555555559ce <+57>:     jne     0x5555555559ea <secret_phase+85>
0x00005555555559d0 <+59>:     lea     0x17a1(%rip),%rdi      # 0x5555555557178
0x00005555555559d7 <+66>:     call   0x555555555200 <puts@plt>
0x00005555555559dc <+71>:     call   0x555555555d6e <phase_defused>
0x00005555555559e1 <+76>:     pop     %rbx
0x00005555555559e2 <+77>:     ret
0x00005555555559e3 <+78>:     call   0x555555555bb5 <explode_bomb>
0x00005555555559e8 <+83>:     jmp     0x5555555559be <secret_phase+41>
0x00005555555559ea <+85>:     call   0x555555555bb5 <explode_bomb>
0x00005555555559ef <+90>:     jmp     0x5555555559d0 <secret_phase+59>
End of assembler dump.
```

```
(gdb) x/d 0x5555555559130
0x5555555559130 <n1>:      36
```

Once checking for <n1> and that's the answer

Secret\_phase answer: 36

## All Answers txt file and giving it input to bomb

```
trillionaire@LAPTOP-VUSF240R:/mnt/c/IIITS ASSIGNMENTS/Sem 2/Computer Arch/Bomb Lab 3$ cat Bomb304_S20230010193_solution.txt
Brownie, you are doing a heck of a job.
1 2 4 7 11 16
1 210
11 18 DrEvil
111116
2 4 6 3 5 1
36
trillionaire@LAPTOP-VUSF240R:/mnt/c/IIITS ASSIGNMENTS/Sem 2/Computer Arch/Bomb Lab 3$ ./bomb Bomb304_S20230010193_solution.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...
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
```