BOMB LAB_1(DOCUMENTATION)

NAME : CHARISHMA BOLLINENI

ROLL NO : S20220010042

SEC : 1

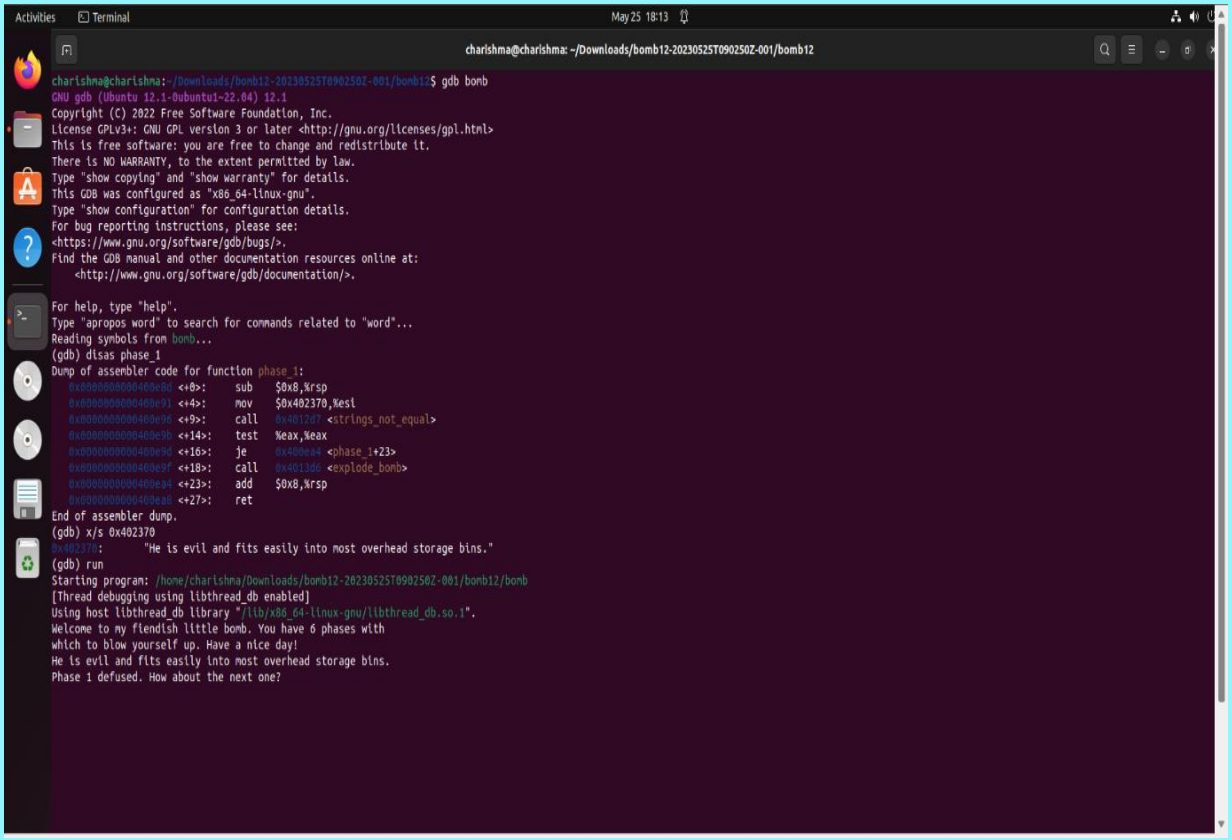
FREQUENTLY USED GDB COMMANDS:

- gdb bomb
- break explode bomb
- break<location>
- run<args>
- disas
- until *<address>
- print $<register>
- kill
- delete

PHASE 1

- Use command “gdb bomb” to start debugger in the terminal.

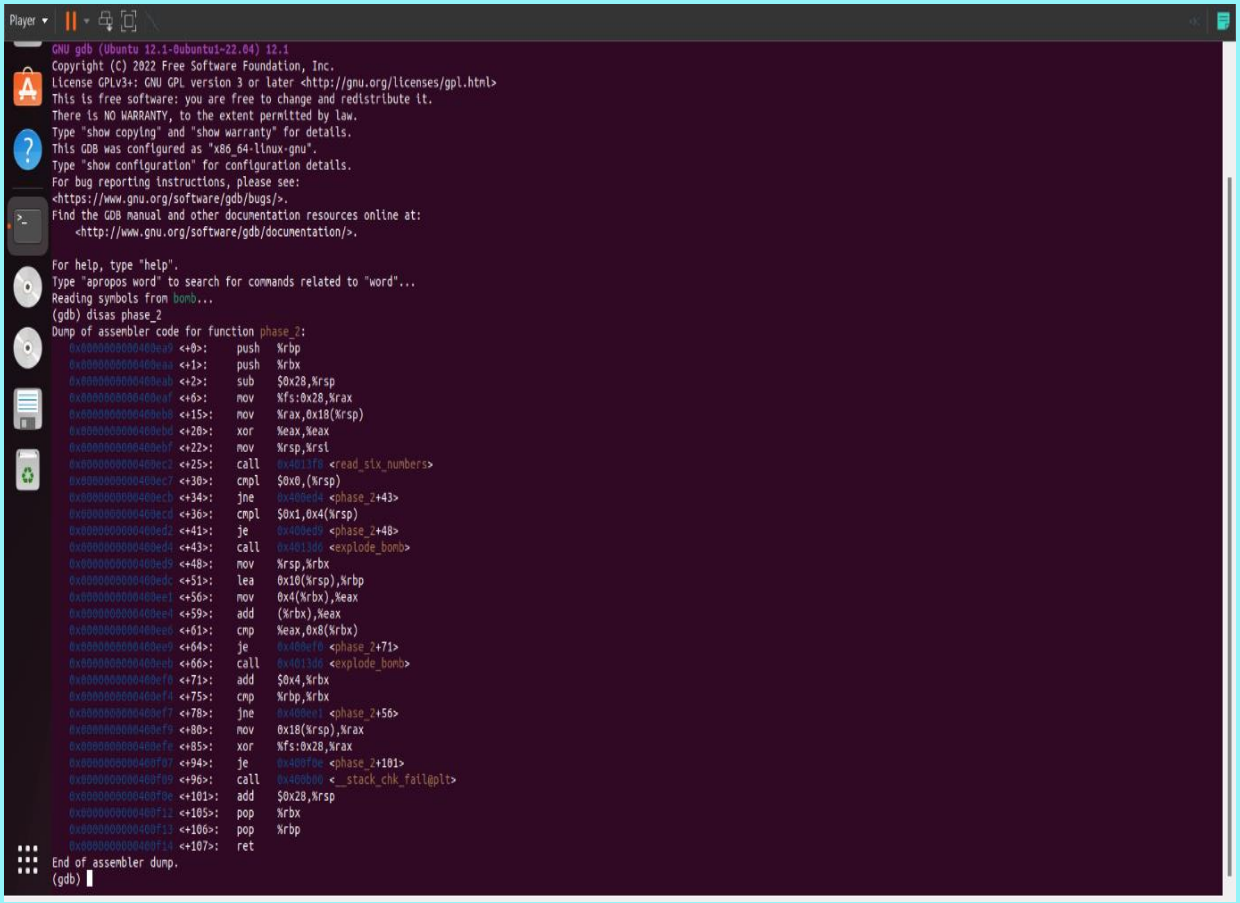
- Now use the command “disas phase_1” to get the dump of assembler code for for function phase_1.
- We can observe that something is being moved into %esi.
- We need to examine what is inside 0x402370, so we use x/s to examine the string inside the address.

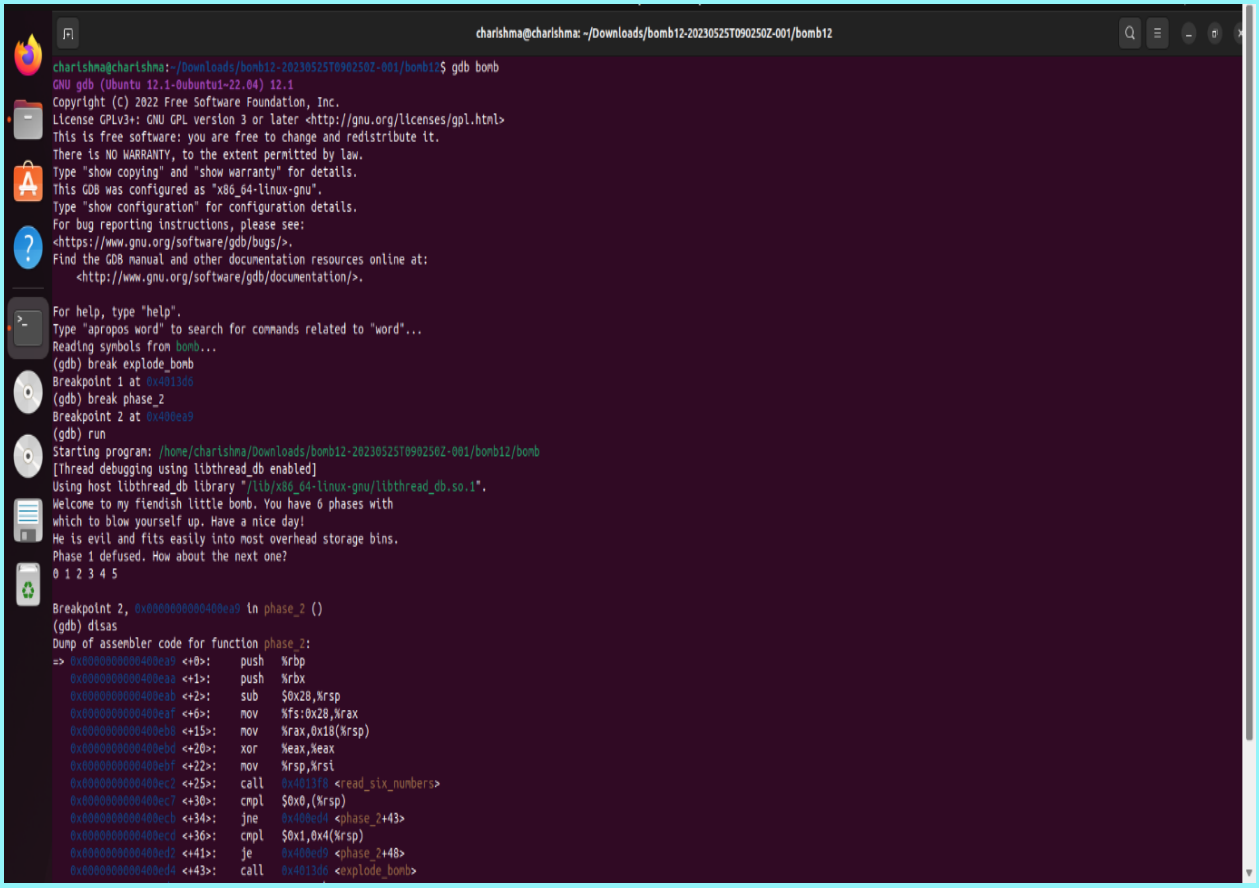


- Hence, we got the solution of Phase_1 (i.e, “ He is an evil and fits easily into most overhead storage bins.”

PHASE 2

- Start the gdb debugger and add break point at phase_2.
- Then add break point at phase_2 using “break” command.
- Now run the program and enter the solution of previous phase.
- Now enter dummy input for phase_2.





- The program will break at phase_2.
- Now use the command “disas phase_2” to get the dump of assembler code for for function phase_2.
- When we Disassemble Function Phase_2, we get this.We can see that it is calling another function read_six_numbers.

charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12

```
charishma@charishma:~/Downloads/bomb12-20230525T090250Z-001/bomb12$ gdb bomb
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 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 bomb...
(gdb) break explode_bomb
Breakpoint 1 at 0x4013d6
(gdb) break phase_2
Breakpoint 2 at 0x400ea9
(gdb) run
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 2 3 4 5

Breakpoint 2, 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>:    call   0x4013f8 <read_six_numbers>
   0x0000000000400ec7 <+30>:    cmpl   $0x0,(%rsp)
   0x0000000000400ecb <+34>:    jne    0x400ed4 <phase_2+43>
   0x0000000000400ecd <+36>:    cmpl   $0x1,0x4(%rsp)
   0x0000000000400ed2 <+41>:    je     0x400ed9 <phase_2+48>
   0x0000000000400ed4 <+43>:    call   0x4013d6 <explode_bomb>
```

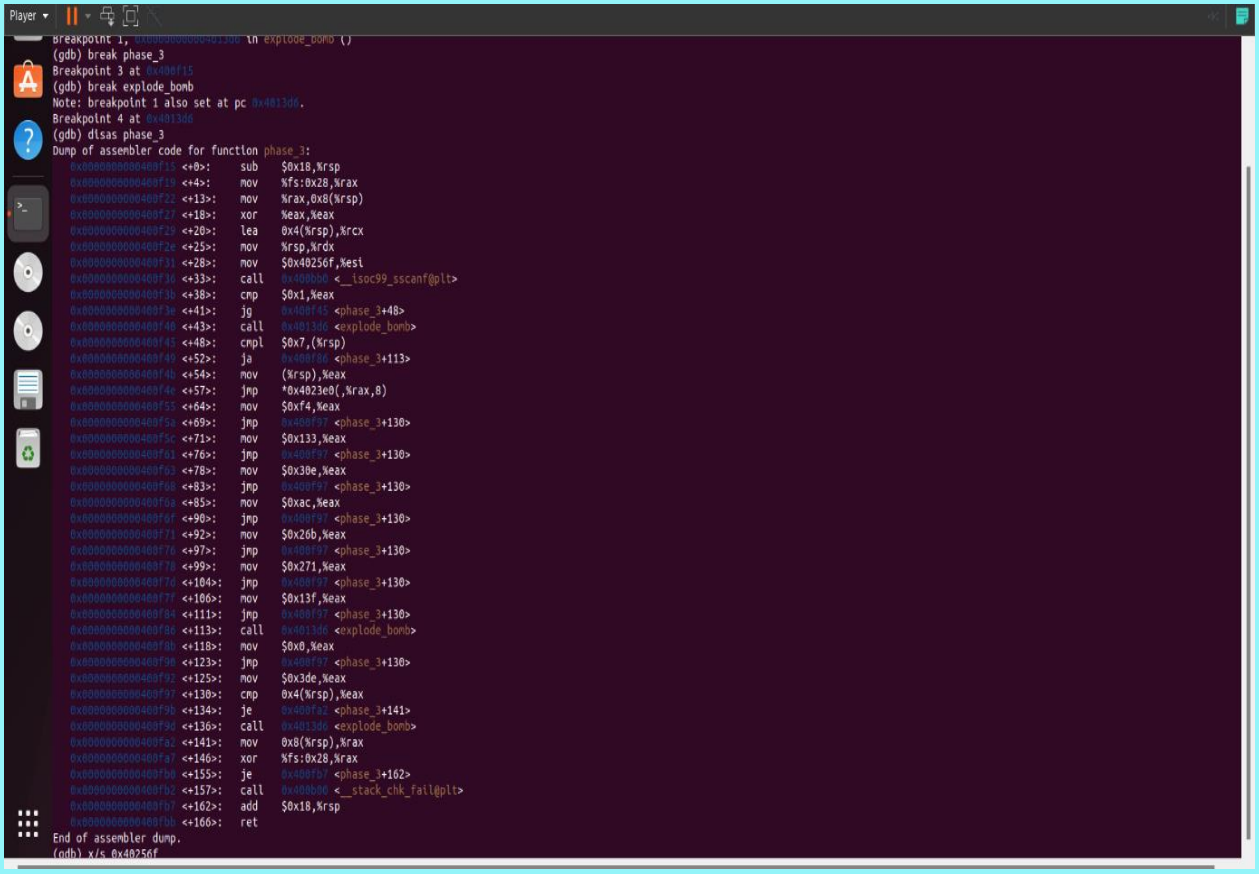
charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12

```
   0x0000000000400ef4 <+75>:    cmp    %rbp,%rbx
   0x0000000000400ef7 <+78>:    jne    0x400ee1 <phase_2+56>
   0x0000000000400ef9 <+80>:    mov    0x18(%rsp),%rax
   0x0000000000400efe <+85>:    xor    %fs:0x28,%rax
   0x0000000000400f07 <+94>:    je     0x400f0e <phase_2+101>
   0x0000000000400f09 <+96>:    call   0x400b00 <__stack_chk_fail@plt>
   0x0000000000400f0e <+101>:   add    $0x28,%rsp
   0x0000000000400f12 <+105>:   pop    %rbx
   0x0000000000400f13 <+106>:   pop    %rbp
   0x0000000000400f14 <+107>:   ret
End of assembler dump.
(gdb) until *0x0000000000400ee6
0x0000000000400ee6 in phase_2 ()
(gdb) print $eax
$7 = 1
(gdb) until *0x0000000000400ee6
0x0000000000400ee6 in phase_2 ()
(gdb) print $eax
$8 = 2
(gdb) until *0x0000000000400ee6
0x0000000000400ee6 in phase_2 ()
(gdb) print $eax
$9 = 3
(gdb) until *0x0000000000400ee6
0x0000000000400ee6 in phase_2 ()
(gdb) print $eax
$10 = 5
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5

Breakpoint 2, 0x0000000000400ea9 in phase_2 ()
(gdb) kill
Kill the program being debugged? (y or n) y
[Inferior 1 (process 3440) killed]
(gdb) delete 2
(gdb) r
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/bomb
[Thread debugging using libthread_db enabled]
```

- We have to give break points at appropriate places such as cmp command,0x0400ee6.Then we check the contents of %eax and %rbx as it is being compared and the result decides whether or not the bomb get exploded.
- So,we check values of registers at this point and verify the value of %eax and (%rbx + 0x8) is same or not.(%eax holds the return value)
- Then we check for the next required number(next iteration).Similarly,we get check the all iterations of the string and get the required number string.
- The required string input is “0 1 1 2 3 5”.

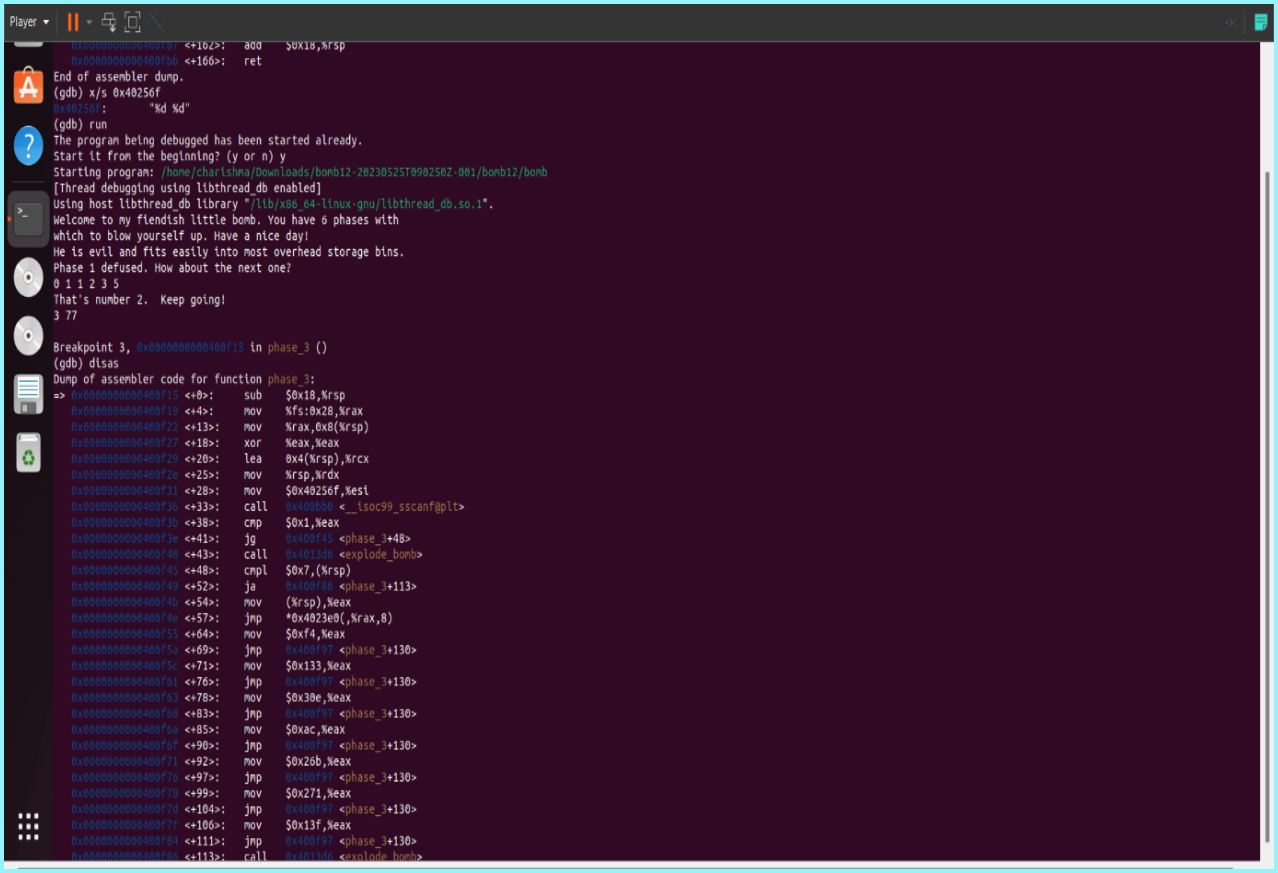
PHASE_3

- When we disassemble function phase_3,we get this.We observe that something is moved into %esi.

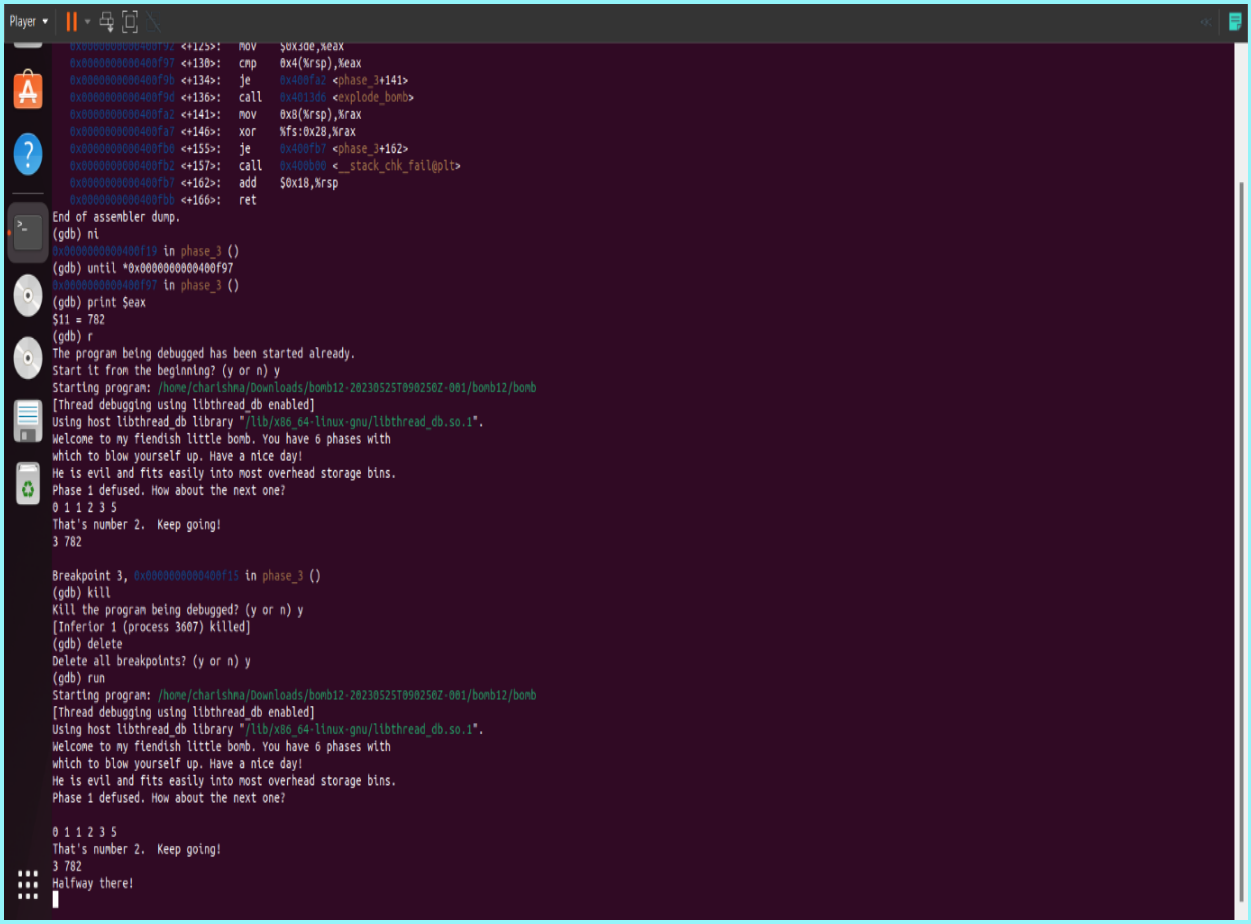


```
Breakpoint 1, 0x0000000000401300 in explode_bomb ()
(gdb) break phase_3
Breakpoint 3 at 0x400f15
(gdb) break explode_bomb
Note: breakpoint 1 also set at pc 0x401300.
Breakpoint 4 at 0x401300
(gdb) disas phase_3
Dump of assembler code for function phase_3:
   0x0000000000400f15 <+0>:     sub    $0x18,%rsp
   0x0000000000400f19 <+4>:     mov    %fs:0x28,%rax
   0x0000000000400f22 <+13>:    mov    %rax,0x8(%rsp)
   0x0000000000400f27 <+18>:    xor    %eax,%eax
   0x0000000000400f29 <+20>:    lea    0x4(%rsp),%rcx
   0x0000000000400f2e <+25>:    mov    %rsp,%rdx
   0x0000000000400f31 <+28>:    mov    $0x40256f,%esi
   0x0000000000400f36 <+33>:    call   0x400bb0 <__isoc99_sscanf@plt>
   0x0000000000400f3b <+38>:    cmp    $0x1,%eax
   0x0000000000400f3e <+41>:    jg     0x400f45 <phase_3+48>
   0x0000000000400f40 <+43>:    call   0x4013d6 <explode_bomb>
   0x0000000000400f45 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f49 <+52>:    ja     0x400f8d <phase_3+113>
   0x0000000000400f4b <+54>:    mov    (%rsp),%eax
   0x0000000000400f4e <+57>:    jmp    *0x4023e0(,%rax,8)
   0x0000000000400f55 <+64>:    mov    $0xf4,%eax
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x133,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x30e,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0xac,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x26b,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x271,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x13f,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   call   0x4013d6 <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3de,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   call   0x4013d6 <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   call   0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   ret
End of assembler dump.
(gdb) x/s 0x40256f
```

- We need to examine what is inside the address 0x40256f,so we examine string inside address.We get to know that it requires a string of 2 integers.
- So we give random inputs to begin.Here, we pass the first explode bomb call as our input string size is correct.

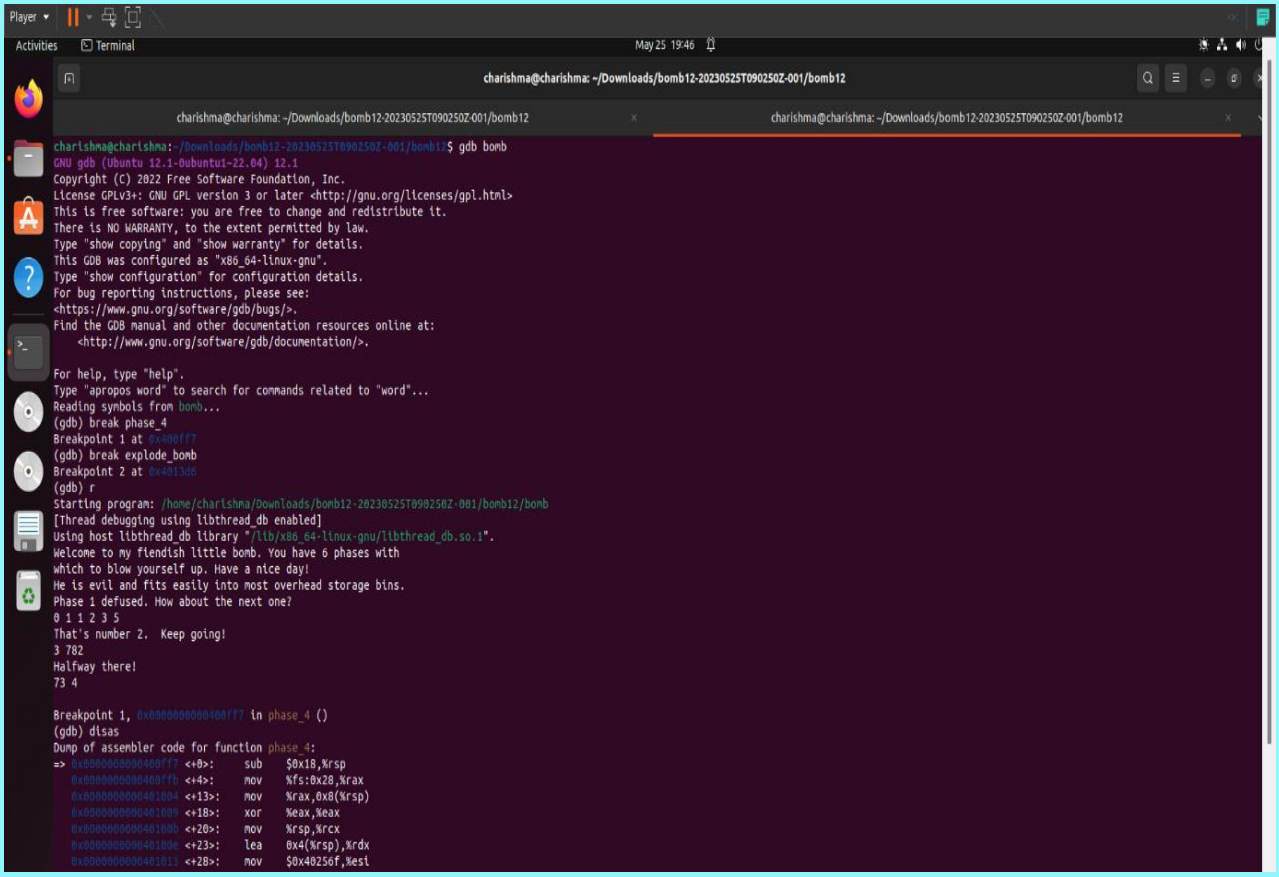


- Then we stop at our next breakpoint (0x400f97).Here,we need to check the contents of %eax and %(rsp + 0x4) as it decides whether or not the bomb get explode.
- Here, we get to know the correct value of our 2nd input.Similarly if give 1st input to know their corresponding 2nd input.
- The required string input can be many but as mine I gave input as “3 782”.

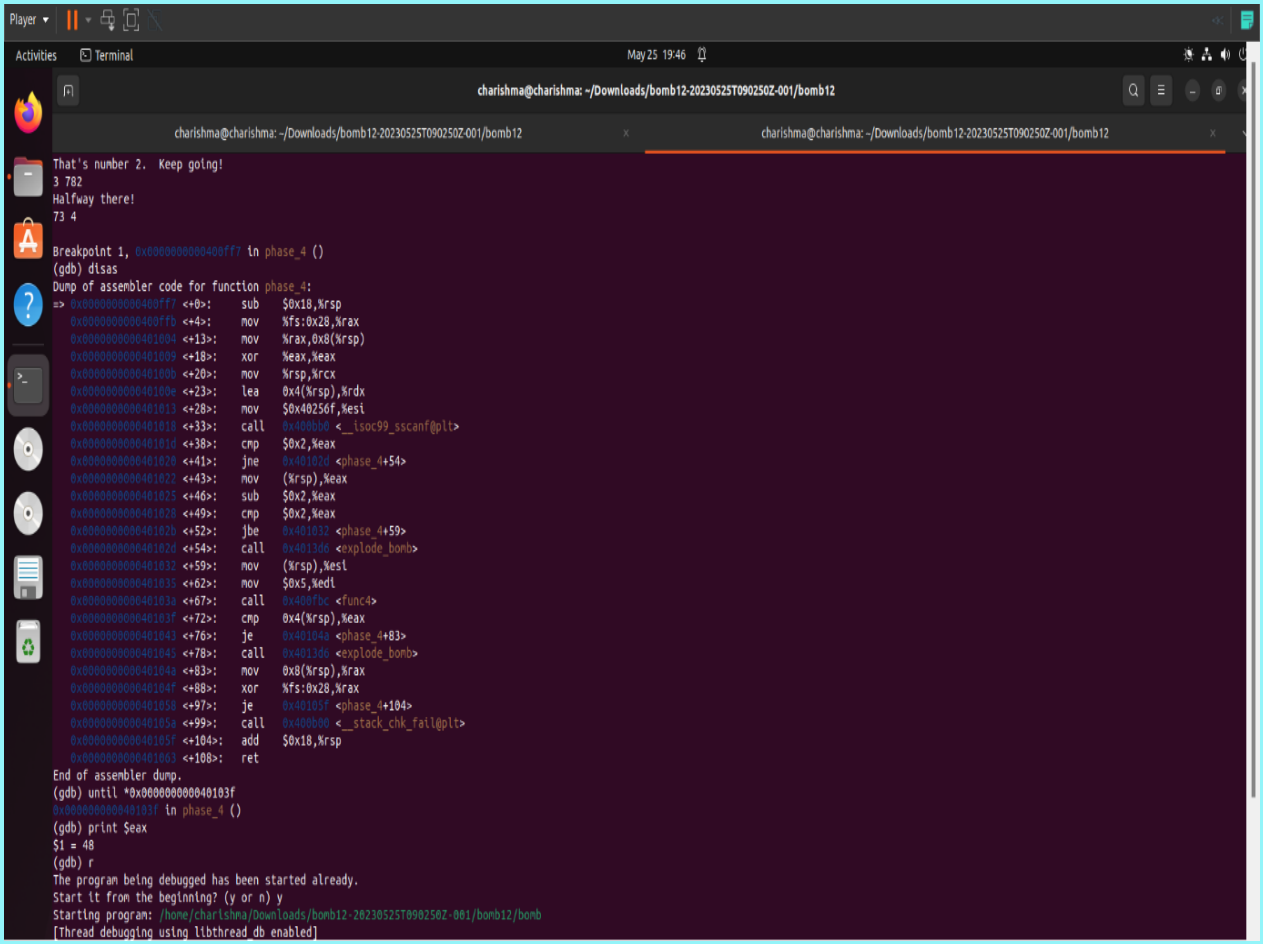


PHASE_4

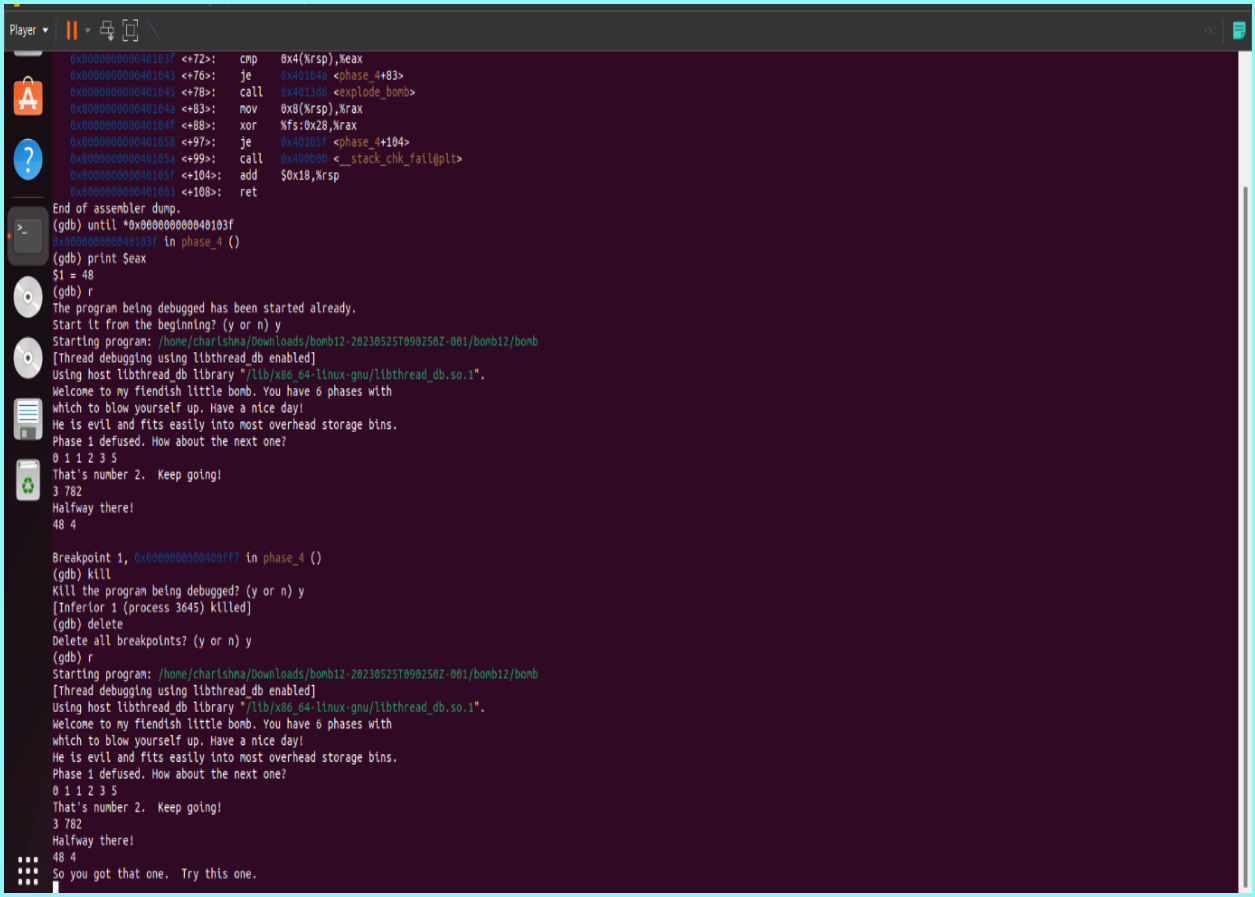
- When we disassemble function phase_4,we get this.We can observe that something is being moved into %esi.We need to examine what’s inside 0x40256f,so we examine string inside the address.
- We get to know that it requires a string of 2integers.So,we give random inputs to begin.



- Here,we passed one explode_bomb call,so our input string is correct.Then according to the assembly code,we can see that after subtracting 2 from our second input it is checking if it is greater than 2 or not.
- This implies that our 2nd input can only be 2,3,4.
- We set our breakpoint at cmp command(0x40103f)to check the contents of registers as they are deciding whether bomb get exploded or not.



- Here,we get to know the 1st input corresponding to value 2.Similarly,we can find corresponding inputs for other 2 numbers.
- The string I got is “48 4”.



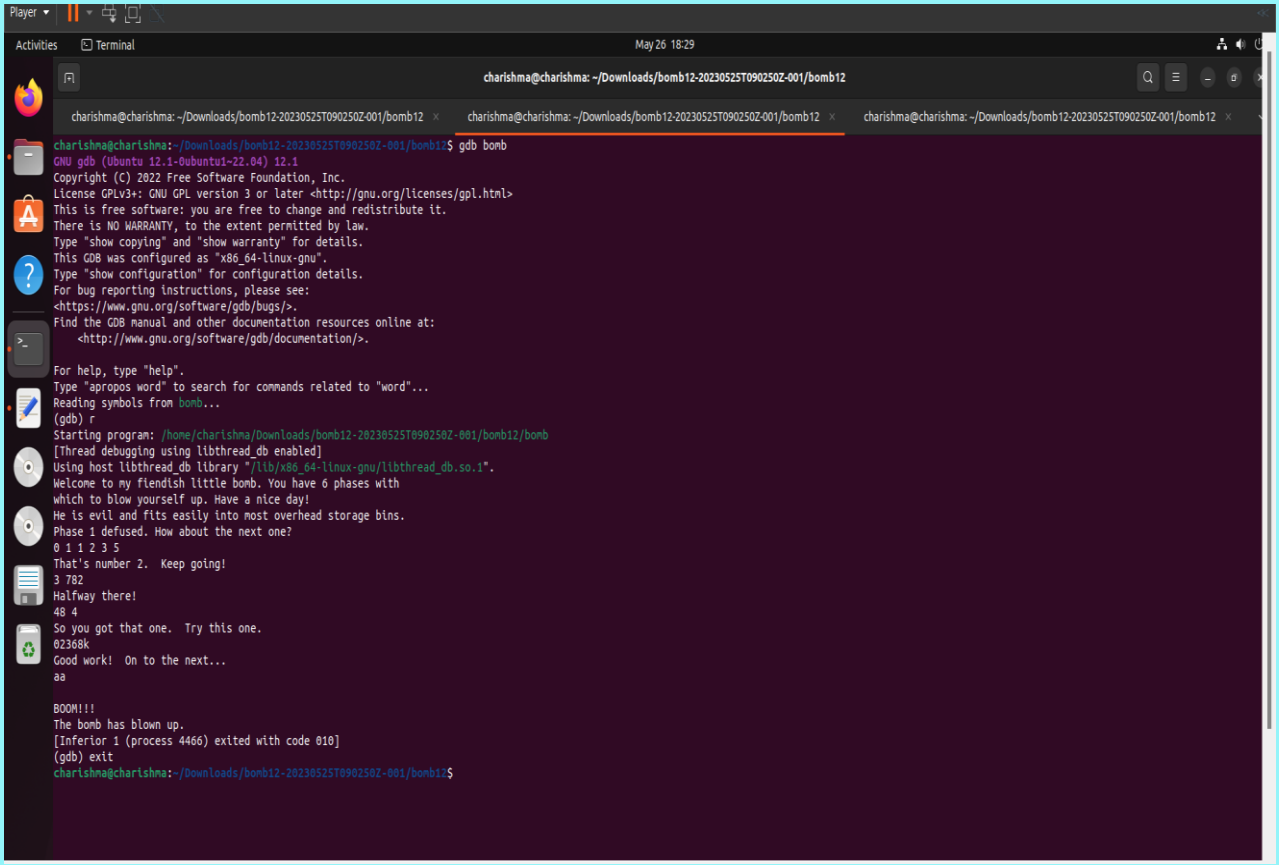
```
Player ▾
   0x000000000040103f <+72>:    cmp    0x4(%rsp),%eax
   0x0000000000401043 <+76>:    je     0x40104a <phase_4+83>
   0x0000000000401045 <+78>:    call   0x4013d6 <explode_bomb>
   0x000000000040104a <+83>:    mov    0x8(%rsp),%rax
   0x000000000040104f <+88>:    xor    %fs:0x28,%rax
   0x0000000000401058 <+97>:    je     0x40105f <phase_4+104>
   0x000000000040105a <+99>:    call   0x400b00 <__stack_chk_fail@plt>
   0x000000000040105f <+104>:   add    $0x18,%rsp
   0x0000000000401063 <+108>:   ret
End of assembler dump.
(gdb) until *0x000000000040103f
0x000000000040103f in phase_4 ()
(gdb) print $eax
$1 = 48
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
48 4

Breakpoint 1, 0x0000000000400ff7 in phase_4 ()
(gdb) kill
Kill the program being debugged? (y or n) y
[Inferior 1 (process 3645) killed]
(gdb) delete
Delete all breakpoints? (y or n) y
(gdb) r
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
48 4
So you got that one.  Try this one.
```

PHASE_5

- When we disassemble function phase_5,we get the following result.
- We see that array present in the register %ecx.Now, we give the indices for the values present in the array starting from ‘0’.
- Now, when we check the %ecx in cmp statement convert 1b into decimal we get to know that the sum of values of indices should be equal to 27.

❖ So, now we need six inputs from the array such that their sum is equal to 27 and later should take their indices as input in reverse order.

```
Breakpoint 1, 0x0000000000401064 in phase_5 ()
(gdb) disas
Dump of assembler code for function phase_5:
=> 0x0000000000401064 <+0>:     push   %rbx
   0x0000000000401065 <+1>:     mov    %rdi,%rbx
   0x0000000000401068 <+4>:     call   0x4012b9 <string_length>
   0x000000000040106d <+9>:     cmp    $0x6,%eax
   0x0000000000401070 <+12>:    je     0x401077 <phase_5+19>
   0x0000000000401072 <+14>:    call   0x4013d6 <explode_bomb>
   0x0000000000401077 <+19>:    mov    %rbx,%rax
   0x000000000040107a <+22>:    lea    0x6(%rbx),%rdi
   0x000000000040107e <+26>:    mov    $0x0,%ecx
   0x0000000000401083 <+31>:    movzbl (%rax),%edx
   0x0000000000401086 <+34>:    and    $0xf,%edx
   0x0000000000401089 <+37>:    add    0x402420(,%rdx,4),%ecx
   0x0000000000401090 <+44>:    add    $0x1,%rax
   0x0000000000401094 <+48>:    cmp    %rdi,%rax
   0x0000000000401097 <+51>:    jne    0x401083 <phase_5+31>
   0x0000000000401099 <+53>:    cmp    $0x1b,%ecx
   0x000000000040109c <+56>:    je     0x4010a3 <phase_5+63>
   0x000000000040109e <+58>:    call   0x4013d6 <explode_bomb>
   0x00000000004010a3 <+63>:    pop    %rbx
   0x00000000004010a4 <+64>:    ret
End of assembler dump.
(gdb) x/16d 0x402420
0x402420 <array.3597>:  2       10      6       1
0x402430 <array.3597+16>:       12      16      9       3
0x402440 <array.3597+32>:       4       7       14      5
0x402450 <array.3597+48>:       11      8       15      13
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
48 4
So you got that one.  Try this one.
12389

Breakpoint 1, 0x0000000000401064 in phase_5 ()
(gdb) kill
Kill the program being debugged? (y or n) y
[Inferior 1 (process 4361) killed]
(gdb) delete
```



- Hence I got solution of Phase_5 as 02368k.

PHASE_6

- When we disassemble Phase_6 function,we get the following result.
- Since it is a linked list we check the addresses of next node and values of them by using x/3x command.
- We do it similarly until we get 6 values and allocate indices to them.Now, have to arrange them in ascending order and give them respective indices.

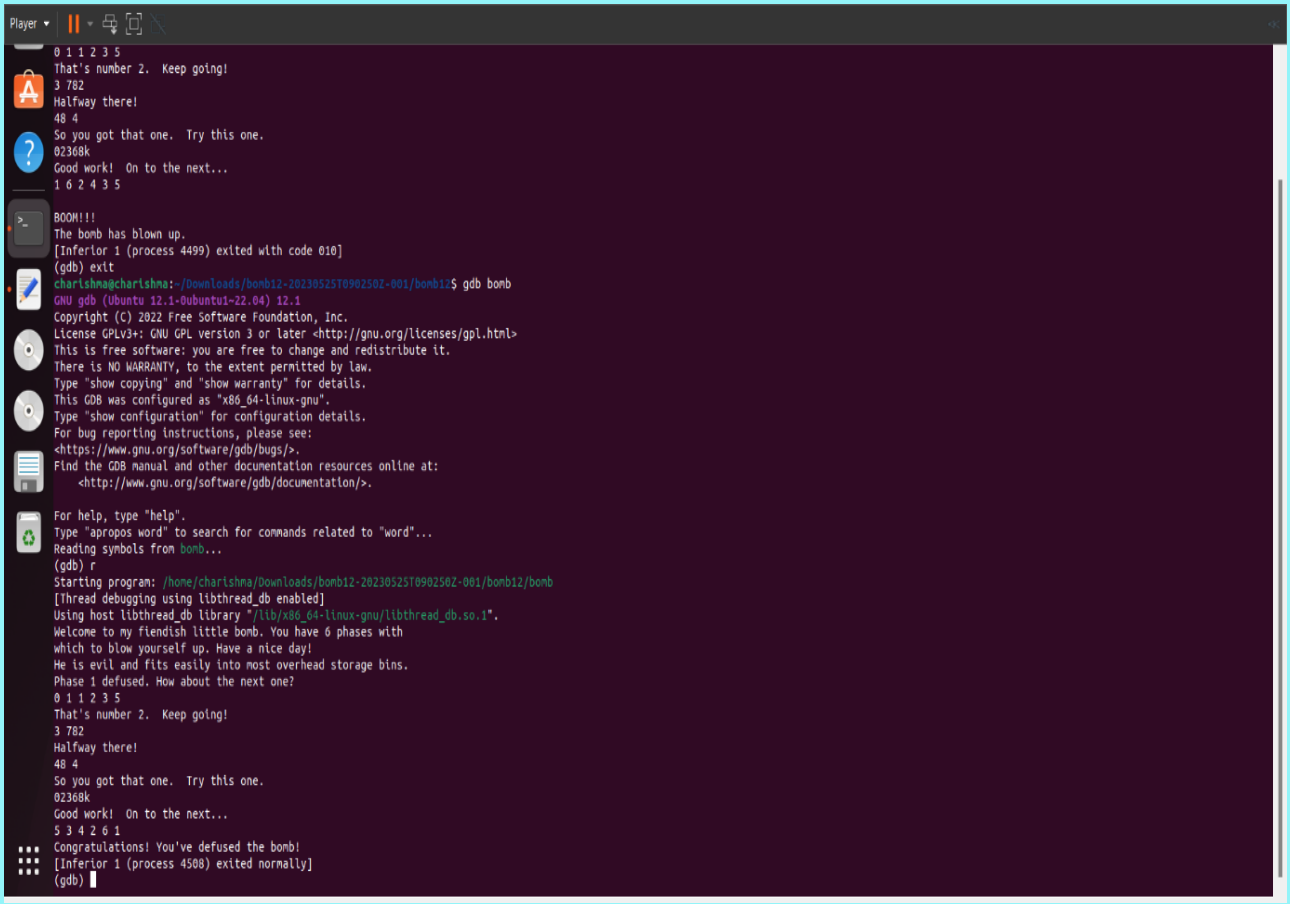
❖ So, we finally give the input(indices).

```
That's number 2.  Keep going!
3 782
Halfway there!
48 4
So you got that one.  Try this one.
02368k
Good work!  On to the next...
aa

Breakpoint 2, 0x00000000004010a5 in phase_6 ()
(gdb) disas
Dump of assembler code for function phase_6:
=> 0x00000000004010a5 <+0>:     push   %r13
   0x00000000004010a7 <+2>:     push   %r12
   0x00000000004010a9 <+4>:     push   %rbp
   0x00000000004010aa <+5>:     push   %rbx
   0x00000000004010ab <+6>:     sub    $0x68,%rsp
   0x00000000004010af <+10>:    mov    %fs:0x28,%rax
   0x00000000004010b8 <+19>:    mov    %rax,0x58(%rsp)
   0x00000000004010bd <+24>:    xor    %eax,%eax
   0x00000000004010bf <+26>:    mov    %rsp,%rsi
   0x00000000004010c2 <+29>:    call   0x4013f8 <read_six_numbers>
   0x00000000004010c7 <+34>:    mov    %rsp,%r12
   0x00000000004010ca <+37>:    mov    $0x0,%r13d
   0x00000000004010d0 <+43>:    mov    %r12,%rbp
   0x00000000004010d3 <+46>:    mov    (%r12),%eax
   0x00000000004010d7 <+50>:    sub    $0x1,%eax
   0x00000000004010da <+53>:    cmp    $0x5,%eax
   0x00000000004010dd <+56>:    jbe    0x4010e4 <phase_6+63>
   0x00000000004010df <+58>:    call   0x4013d6 <explode_bomb>
   0x00000000004010e4 <+63>:    add    $0x1,%r13d
   0x00000000004010e8 <+67>:    cmp    $0x6,%r13d
   0x00000000004010ec <+71>:    je     0x40112b <phase_6+134>
   0x00000000004010ee <+73>:    mov    %r13d,%ebx
   0x00000000004010f1 <+76>:    movslq %ebx,%rax
   0x00000000004010f4 <+79>:    mov    (%rsp,%rax,4),%eax
   0x00000000004010f7 <+82>:    cmp    %eax,0x0(%rbp)
   0x00000000004010fa <+85>:    jne    0x401101 <phase_6+92>
   0x00000000004010fc <+87>:    call   0x4013d6 <explode_bomb>
   0x0000000000401101 <+92>:    add    $0x1,%ebx
   0x0000000000401104 <+95>:    cmp    $0x5,%ebx
   0x0000000000401107 <+98>:    jle    0x4010f1 <phase_6+76>
   0x0000000000401109 <+100>:   add    $0x4,%r12
   0x000000000040110d <+104>:   jmp    0x4010d0 <phase_6+43>
   0x000000000040110f <+106>:   mov    0x8(%rdx),%rdx
```

```
   0x00000000004010e4 <+63>:    add    $0x1,%r13d
   0x00000000004010e8 <+67>:    cmp    $0x6,%r13d
   0x00000000004010ec <+71>:    je     0x40112b <phase_6+134>
   0x00000000004010ee <+73>:    mov    %r13d,%ebx
   0x00000000004010f1 <+76>:    movslq %ebx,%rax
   0x00000000004010f4 <+79>:    mov    (%rsp,%rax,4),%eax
   0x00000000004010f7 <+82>:    cmp    %eax,0x0(%rbp)
   0x00000000004010fa <+85>:    jne    0x401101 <phase_6+92>
   0x00000000004010fc <+87>:    call   0x4013d6 <explode_bomb>
   0x0000000000401101 <+92>:    add    $0x1,%ebx
   0x0000000000401104 <+95>:    cmp    $0x5,%ebx
   0x0000000000401107 <+98>:    jle    0x4010f1 <phase_6+76>
   0x0000000000401109 <+100>:   add    $0x4,%r12
   0x000000000040110d <+104>:   jmp    0x4010d0 <phase_6+43>
   0x000000000040110f <+106>:   mov    0x8(%rdx),%rdx
   0x0000000000401113 <+110>:   add    $0x1,%eax
   0x0000000000401116 <+113>:   cmp    %ecx,%eax
   0x0000000000401118 <+115>:   jne    0x40110f <phase_6+106>
   0x000000000040111a <+117>:   mov    %rdx,0x20(%rsp,%rsi,2)
   0x000000000040111f <+122>:   add    $0x4,%rsi
   0x0000000000401123 <+126>:   cmp    $0x18,%rsi
   0x0000000000401127 <+130>:   jne    0x401130 <phase_6+139>
   0x0000000000401129 <+132>:   jmp    0x401144 <phase_6+159>
   0x000000000040112b <+134>:   mov    $0x0,%esi
   0x0000000000401130 <+139>:   mov    (%rsp,%rsi,1),%ecx
   0x0000000000401133 <+142>:   mov    $0x1,%eax
   0x0000000000401138 <+147>:   mov    $0x6032f0,%edx
   0x000000000040113d <+152>:   cmp    $0x1,%ecx
   0x0000000000401140 <+155>:   jg     0x40110f <phase_6+106>
   0x0000000000401142 <+157>:   jmp    0x40111a <phase_6+117>
   0x0000000000401144 <+159>:   mov    0x20(%rsp),%rbx
   0x0000000000401149 <+164>:   lea    0x20(%rsp),%rax
   0x000000000040114e <+169>:   lea    0x48(%rsp),%rsi
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000000000401153 <+174>:   mov    %rbx,%rcx
   0x0000000000401156 <+177>:   mov    0x8(%rax),%rdx
   0x000000000040115a <+181>:   mov    %rdx,0x8(%rcx)
   0x000000000040115e <+185>:   add    $0x8,%rax
   0x0000000000401162 <+189>:   mov    %rdx,%rcx
   0x0000000000401165 <+192>:   cmp    %rsi,%rax
   0x0000000000401168 <+195>:   jne    0x401156 <phase_6+177>
   0x000000000040116a <+197>:   movq   $0x0,0x8(%rdx)
   0x0000000000401172 <+205>:   mov    $0x5,%ebp
   0x0000000000401177 <+210>:   mov    0x8(%rbx),%rax
   0x000000000040117b <+214>:   mov    (%rax),%eax
```

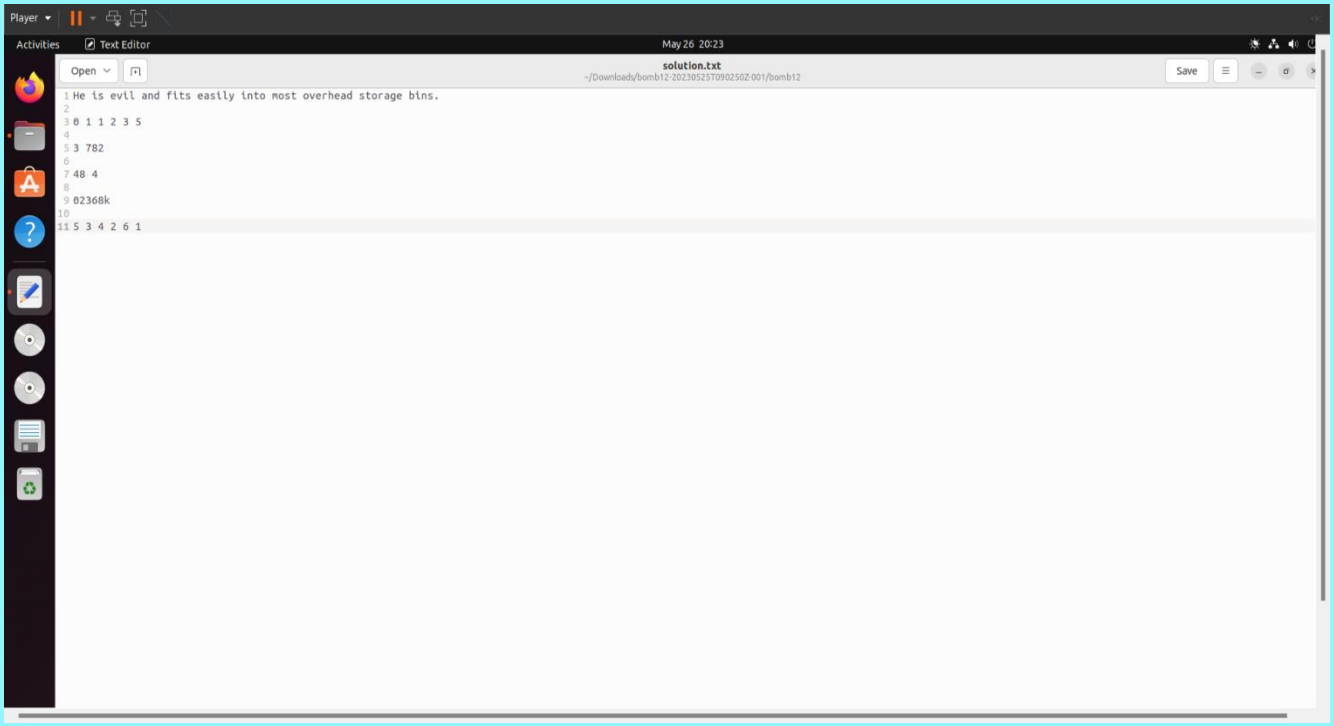
```
Player
Activities  Terminal                                   May 26 18:33
charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12
charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12   charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12   charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12

   0x000000000040115a <+181>:   mov    %rdx,0x8(%rcx)
   0x000000000040115e <+185>:   add    $0x8,%rax
   0x0000000000401162 <+189>:   mov    %rdx,%rcx
   0x0000000000401165 <+192>:   cmp    %rsi,%rax
   0x0000000000401168 <+195>:   jne    0x401150 <phase_6+177>
   0x000000000040116a <+197>:   movq   $0x0,0x8(%rdx)
   0x0000000000401172 <+205>:   mov    $0x5,%ebp
   0x0000000000401177 <+210>:   mov    0x8(%rbx),%rax
   0x000000000040117b <+214>:   mov    (%rax),%eax
   0x000000000040117d <+216>:   cmp    %eax,(%rbx)
   0x000000000040117f <+218>:   jge    0x401186 <phase_6+225>
   0x0000000000401181 <+220>:   call   0x4013d0 <explode_bomb>
   0x0000000000401186 <+225>:   mov    0x8(%rbx),%rbx
   0x000000000040118a <+229>:   sub    $0x1,%ebp
   0x000000000040118d <+232>:   jne    0x401177 <phase_6+210>
   0x000000000040118f <+234>:   mov    0x58(%rsp),%rax
   0x0000000000401194 <+239>:   xor    %fs:0x28,%rax
   0x000000000040119d <+248>:   je     0x4011a4 <phase_6+255>
   0x000000000040119f <+250>:   call   0x400b80 <__stack_chk_fail@plt>
   0x00000000004011a4 <+255>:   add    $0x68,%rsp
   0x00000000004011a8 <+259>:   pop    %rbx
   0x00000000004011a9 <+260>:   pop    %rbp
   0x00000000004011aa <+261>:   pop    %r12
   0x00000000004011ac <+263>:   pop    %r13
   0x00000000004011ae <+265>:   ret
End of assembler dump.
(gdb) x/3x 0x40110f
0x40110f <phase_6+106>: 0x08528b48      0x3901c083      0x48f575c8
(gdb) x/3x  0x6032f0
0x6032f0 <node1>:       0x0000007f      0x00000001      0x00603300
(gdb) p 0x0000007f
$1 = 127
(gdb) x/3x 0x00603300
0x603300 <node2>:       0x000001f3      0x00000002      0x00603310
(gdb) Quit
(gdb) p 0x00603300
$2 = 6304512
(gdb) p 0x000001f3
$3 = 499
(gdb) x/3x 0x00603310
0x603310 <node3>:       0x00000336      0x00000003      0x00603320
(gdb) p 0x00000336
$4 = 822
(gdb) x/3x 0x00603320
0x603320 <node4>:       0x0000031e      0x00000004      0x00603330
```

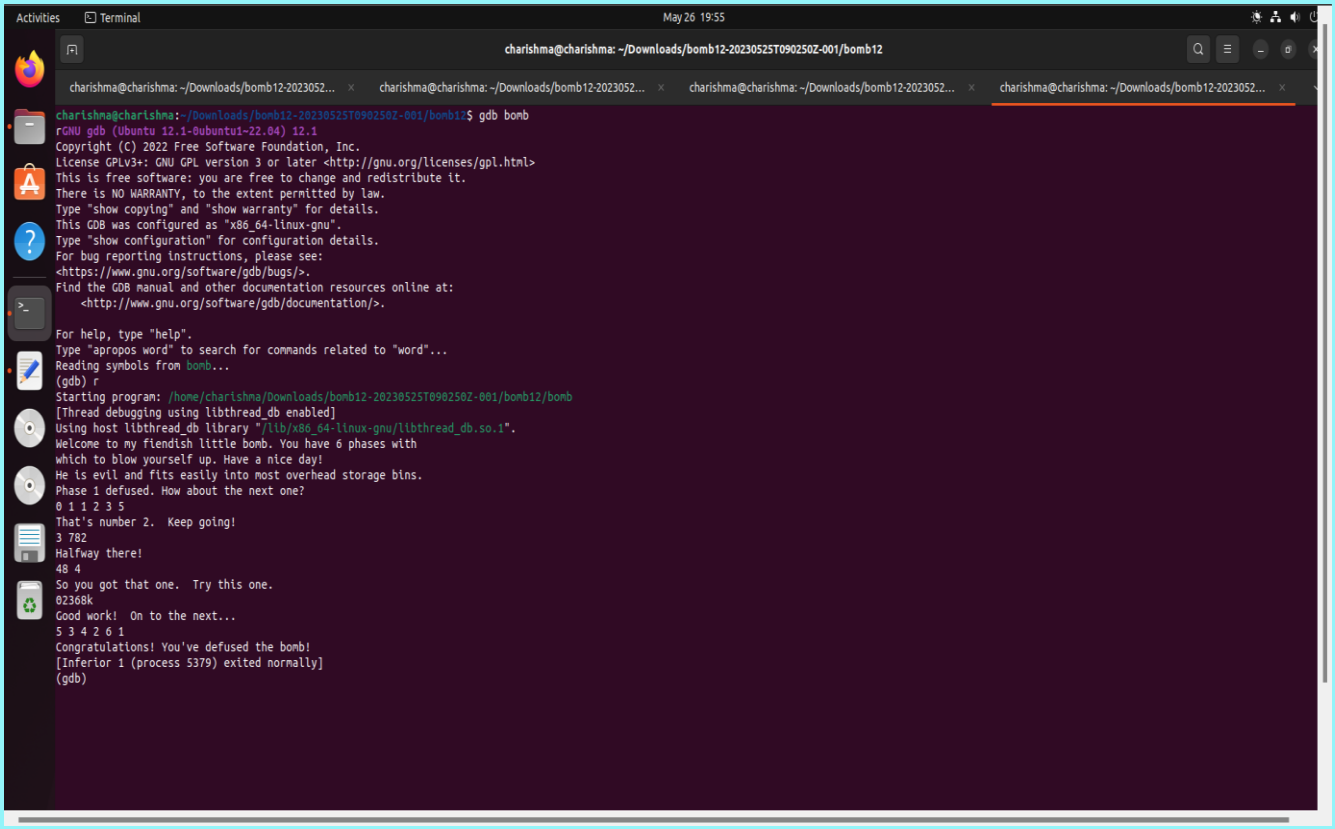
```
Player
Activities  Terminal                                   May 26 18:33
charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12
charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12   charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12   charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12

He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
48 4
So you got that one.  Try this one.
02368k
Good work!  On to the next...
1 6 2 4 3 5

BOOM!!!
The bomb has blown up.
[Inferior 1 (process 4499) exited with code 010]
(gdb) exit
charishma@charishma:~/Downloads/bomb12-20230525T090250Z-001/bomb12$ gdb bomb
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 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 bomb...
(gdb) r
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
```



```
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
48 4
So you got that one.  Try this one.
02368k
Good work!  On to the next...
1 6 2 4 3 5

BOOM!!!
The bomb has blown up.
[Inferior 1 (process 4499) exited with code 010]
(gdb) exit
charishma@charishma:~/Downloads/bomb12-20230525T090250Z-001/bomb12$ gdb bomb
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 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 bomb...
(gdb) r
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 782
Halfway there!
48 4
So you got that one.  Try this one.
02368k
Good work!  On to the next...
5 3 4 2 6 1
Congratulations! You've defused the bomb!
[Inferior 1 (process 4508) exited normally]
(gdb)
```

❖ Hence, I got my solution as 5 3 4 2 6 1.


Player
Activities
Text Editor
May 26 20:23
Open
solution.txt
~/Downloads/bomb12-20230525T090250Z-001/bomb12
Save
He is evil and fits easily into most overhead storage bins.
0 1 1 2 3 5
3 782
48 4
02368k
5 3 4 2 6 1


Activities
Terminal
May 26 19:55
charishma@charishma: ~/Downloads/bomb12-20230525T090250Z-001/bomb12
charishma@charishma: ~/Downloads/bomb12-2023052...
charishma@charishma: ~/Downloads/bomb12-2023052...
charishma@charishma: ~/Downloads/bomb12-2023052...
charishma@charishma: ~/Downloads/bomb12-2023052...
charishma@charishma:~/Downloads/bomb12-20230525T090250Z-001/bomb12$ gdb bomb
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 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 bomb...
(gdb) r
Starting program: /home/charishma/Downloads/bomb12-20230525T090250Z-001/bomb12/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!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2. Keep going!
3 782
Halfway there!
48 4
So you got that one. Try this one.
02368k
Good work! On to the next...
5 3 4 2 6 1
Congratulations! You've defused the bomb!
[Inferior 1 (process 5379) exited normally]
(gdb)

THANK YOU….