Phase 3 is about giving two integer in which If the two integers are correct than the phase_3 bomb will be defuse. For that:

The procedure are all same where we have to start with gdb bomb and give break point for phase_3 and give answer for phase 1 and 2 and in third we can give any random two integers for now.

```
Activities   Terminal                    20 10:33                                   en
rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003
(gdb) u * 0x0000000000400f36
0x0000000000400f36 in phase_3 ()
(gdb) disas
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    $0x4025af,%esi
=> 0x0000000000400f36 <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x0000000000400f3b <+38>:    cmp    $0x1,%eax
   0x0000000000400f3e <+41>:    jg     0x400f45 <phase_3+48>
   0x0000000000400f40 <+43>:    callq  0x40142a <explode_bomb>
   0x0000000000400f45 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f49 <+52>:    ja     0x400f86 <phase_3+113>
   0x0000000000400f4b <+54>:    mov    (%rsp),%eax
   0x0000000000400f4e <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f55 <+64>:    mov    $0xc6,%eax
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
--Type <RET> for more, q to quit, c to continue without paging--
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) x/s0x4025af
0x4025af:       "%d %d"
(gdb) r ans.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/rinchen/Downloads/Assignment 1_2/Assignment 1/bomb003/bomb ans.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```

Out here we have to look at line <33> where there is callq function and it calls "scanf" where we can look at the line above it at line <28> in which we can see that there is one hex address. So we can write "x/s the hex number" which will help us getting how much input is required in phase_3 that is two integers. As found that the inputs are two integers as there is two ""%d" "%d"".

Now lets go on and see the comparison code in the line <33> where we have to compare (1 and %eax).From there we can move to line <38> by commanding “**until* address**” , after that we don’t know the value of eax so we can find it with the help of information register.

```
0x0000000000400f3b in phase_3 ()
(gdb) disas
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    $0x4025af,%esi
   0x0000000000400f36 <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
=> 0x0000000000400f3b <+38>:    cmp    $0x1,%eax
   0x0000000000400f3e <+41>:    jg     0x400f45 <phase_3+48>
   0x0000000000400f40 <+43>:    callq  0x40142a <explode_bomb>
   0x0000000000400f45 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f49 <+52>:    ja     0x400f86 <phase_3+113>
   0x0000000000400f4b <+54>:    mov    (%rsp),%eax
   0x0000000000400f4e <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f55 <+64>:    mov    $0xc6,%eax
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
--Type <RET> for more, q to quit, c to continue without paging--
```

In line <38> it compares (1 and %eax) and the line below that states a function which says jg(it is unsigned and it jump if greater than). We can figure out that if the input is less than 1 the bomb will get blast. So the first digit should be greater than or equal to 1. After that it will move to line <41> and call the “jg” function. After that the jg function will move to line <48> where it compares (7 and %rsp) .Out here, if the value of rsp is greater than 7 than the function will move to the line<113> and the bomb will get blast. From here we can know that the first input should be greater than or equal to 1 and less than or equal to 7.

Lets us run the program and give the first input as any number from range (1-7) since the number should be greater than or equal to 1 and less than or equal to 7.

```
Activities   Terminal                                                    en
rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) x/s0x4025af
0x4025af:       "%d %d"
(gdb) r ans.txt
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/rinchen/Downloads/Assignment 1_2/Assignment 1/bomb003/bomb ans.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?
0 1 1 2 3 5
That's number 2.  Keep going!
2 888

Breakpoint 1, 0x0000000000400f15 in phase_3 ()
(gdb) disas
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    $0x4025af,%esi
   0x0000000000400f36 <+33>:    callq  0x400bb0 <__isoc99_sscanf@plt>
   0x0000000000400f3b <+38>:    cmp    $0x1,%eax
   0x0000000000400f3e <+41>:    jg     0x400f45 <phase_3+48>
   0x0000000000400f40 <+43>:    callq  0x40142a <explode_bomb>
   0x0000000000400f45 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f49 <+52>:    ja     0x400f86 <phase_3+113>
   0x0000000000400f4b <+54>:    mov    (%rsp),%eax
   0x0000000000400f4e <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f55 <+64>:    mov    $0xc6,%eax
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
```

We still have to find the second integer and we need to go to the first compare function at line <38> from there it will compare the given user input integer with 1. If it is greater than or equal to 1 it will move to the next function which is jg(jump if greater than or equals to) at line<41>.From there it will move to line <48>. There we have to compare our input with 7, where it will check if the number is less than or equal to 7. And if the number is greater than 7 it will go to next line and the boob will get blast. But as our input is less than 7 so it will move to line <54>. Now we will now go to the next compare function at line <+130> and now lets check there value in information register we get the second integer input, since the second input is be stored in the %eax register.

```
Activities   Terminal                                  20 10:34                                   en
rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003
   0x0000000000400f4e <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f55 <+64>:    mov    $0xc6,%eax
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
=> 0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) i r
rax            0x31e               798
rbx            0x7fffffffded8      140737488346840
rcx            0x0                 0
rdx            0x7fffffffddc4      140737488346564
rsi            0x0                 0
rdi            0x7fffffffd770      140737488344944
rbp            0x0                 0x0
rsp            0x7fffffffddc0      0x7fffffffddc0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f59ac0      140737353456320
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffded0      140737488346832
r14            0x0                 0
r15            0x0                 0
rip            0x400f97            0x400f97 <phase_3+130>
eflags         0x293               [ CF AF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

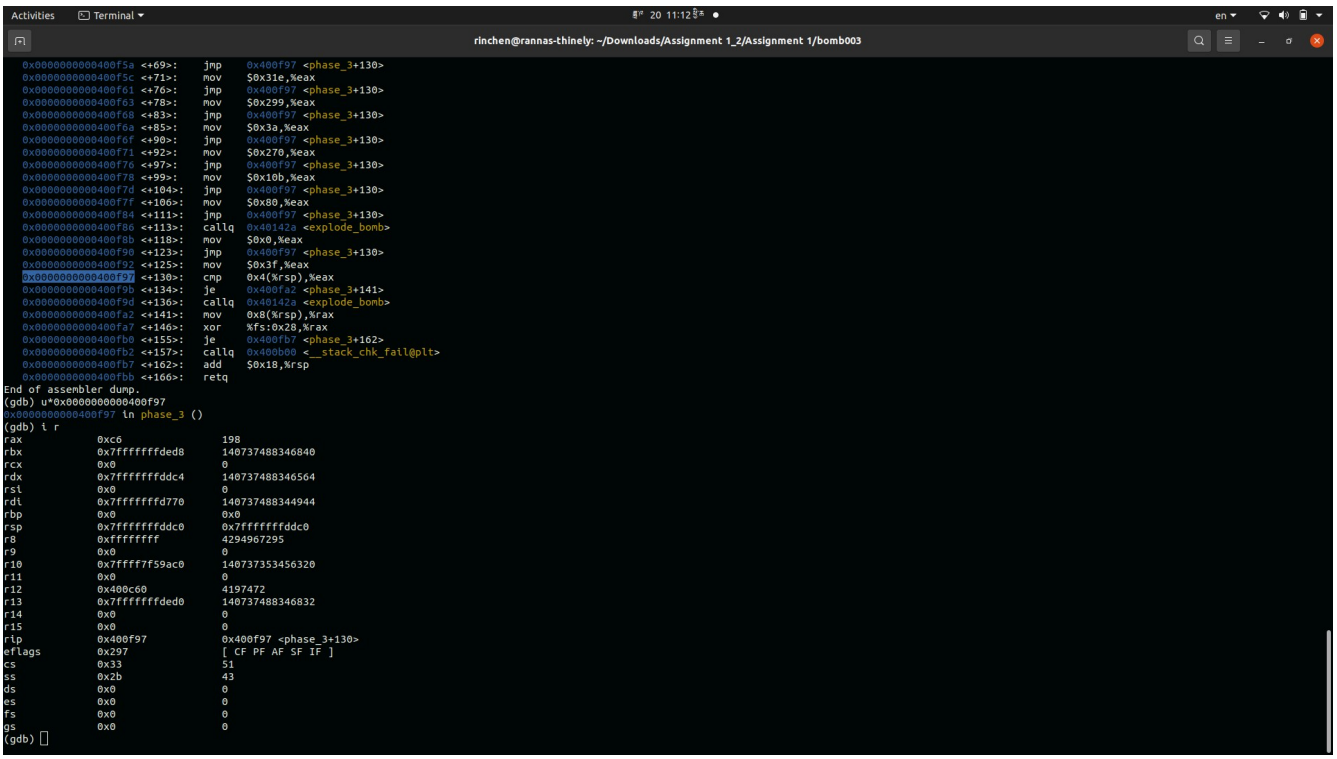
Here in above picture we can see that for the first input 2 we get the second input as 798.

```
.eh_frame
.init_array
.fini_array
.jcr
.dynamic
.got.plt
.data
.bss
.comment
.debug_aranges
.debug_info
.debug_abbrev
.debug_line
.debug_str
.debug_loc
rinchen@rannas-thinely:~/Downloads/Assignment 1_2/Assignment 1/bomb003$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
2 798
Halfway there!
^[[A
```

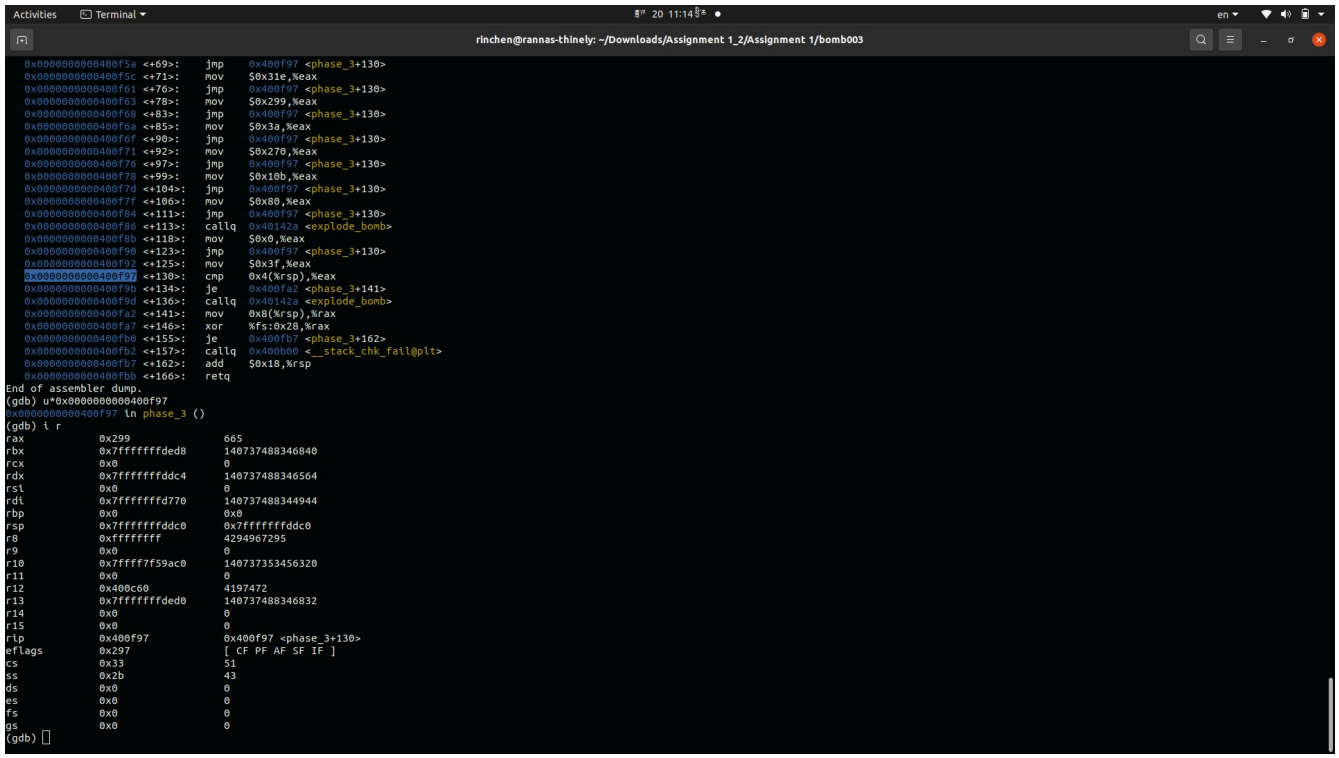
when we try to put 2 and 798 to see if the bomb will be diffused or not.

In phase_3 it accepts the first inputs in the range of (1-7) so therefore we can give any first input in the range (1-7) in which there will be different first input, there will be different second input.

For First input 1 The second input is 198



For First input 3 The second input is 655



For First input 4 The second input is 58

```
Activities   Terminal                                  20 11:15                                   en
rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) u*0x0000000000400f97
0x0000000000400f97 in phase_3 ()
(gdb) i r
rax            0x3a                58
rbx            0x7fffffffded8      140737488346840
rcx            0x0                 0
rdx            0x7fffffffddc4      140737488346564
rsi            0x0                 0
rdi            0x7fffffffd770      140737488344944
rbp            0x0                 0x0
rsp            0x7fffffffddc0      0x7fffffffddc0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f59ac0      140737353456320
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffded0      140737488346832
r14            0x0                 0
r15            0x0                 0
rip            0x400f97            0x400f97 <phase_3+130>
eflags         0x293               [ CF AF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

For First input 5 The second input is 624

```
Activities   Terminal                                  20 11:17                                   en
rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) u*0x0000000000400f97
0x0000000000400f97 in phase_3 ()
(gdb) i r
rax            0x270               624
rbx            0x7fffffffded8      140737488346840
rcx            0x0                 0
rdx            0x7fffffffddc4      140737488346564
rsi            0x0                 0
rdi            0x7fffffffd770      140737488344944
rbp            0x0                 0x0
rsp            0x7fffffffddc0      0x7fffffffddc0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f59ac0      140737353456320
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffded0      140737488346832
r14            0x0                 0
r15            0x0                 0
rip            0x400f97            0x400f97 <phase_3+130>
eflags         0x293               [ CF AF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

For First input 6 The second input is 267

```
Activities   Terminal                                  20 11:18                                   en
rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) u*0x0000000000400f97
0x0000000000400f97 in phase_3 ()
(gdb) i r
rax            0x10b               267
rbx            0x7fffffffded8      140737488346840
rcx            0x0                 0
rdx            0x7fffffffddc4      140737488346564
rsi            0x0                 0
rdi            0x7fffffffd770      140737488344944
rbp            0x0                 0x0
rsp            0x7fffffffddc0      0x7fffffffddc0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f59ac0      140737353456320
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffded0      140737488346832
r14            0x0                 0
r15            0x0                 0
rip            0x400f97            0x400f97 <phase_3+130>
eflags         0x297               [ CF PF AF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

For First input 7 The second input is 128

Activities Terminal 月 20 11:19 ● en

rinchen@rannas-thinely: ~/Downloads/Assignment 1_2/Assignment 1/bomb003

```
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
   0x0000000000400f61 <+76>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f63 <+78>:    mov    $0x299,%eax
   0x0000000000400f68 <+83>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f6a <+85>:    mov    $0x3a,%eax
   0x0000000000400f6f <+90>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f71 <+92>:    mov    $0x270,%eax
   0x0000000000400f76 <+97>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f78 <+99>:    mov    $0x10b,%eax
   0x0000000000400f7d <+104>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f7f <+106>:   mov    $0x80,%eax
   0x0000000000400f84 <+111>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f86 <+113>:   callq  0x40142a <explode_bomb>
   0x0000000000400f8b <+118>:   mov    $0x0,%eax
   0x0000000000400f90 <+123>:   jmp    0x400f97 <phase_3+130>
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
   0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
   0x0000000000400fb0 <+155>:   je     0x400fb7 <phase_3+162>
   0x0000000000400fb2 <+157>:   callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400fb7 <+162>:   add    $0x18,%rsp
   0x0000000000400fbb <+166>:   retq
End of assembler dump.
(gdb) u*0x0000000000400f97
0x0000000000400f97 in phase_3 ()
(gdb) i r
rax            0x80                128
rbx            0x7fffffffded8      140737488346840
rcx            0x0                 0
rdx            0x7fffffffddc4      140737488346564
rsi            0x0                 0
rdl            0x7fffffffd770      140737488344944
rbp            0x0                 0x0
rsp            0x7fffffffddc0      0x7fffffffddc0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f59ac0      140737353456320
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffded0      140737488346832
r14            0x0                 0
r15            0x0                 0
rip            0x400f97            0x400f97 <phase_3+130>
eflags         0x246               [ PF ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```