Name :-

Purval Madhukar Bhude

Roll No. S20230010193

Subject :- CA

Bomb Lab

Phase 1:

```
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 2/PracticeBomb1/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!
sda

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   0x555555555baf <strings_not_equal>
   0x00005555555555bb <+20>:    test   %eax,%eax
   0x00005555555555bd <+22>:    jne    0x5555555555c4 <phase_1+29>
   0x00005555555555bf <+24>:    add    $0x8,%rsp
   0x00005555555555c3 <+28>:    ret
   0x00005555555555c4 <+29>:    call   0x555555555cc3 <explode_bomb>
   0x00005555555555c9 <+34>:    jmp    0x5555555555bf <phase_1+24>
End of assembler dump.
(gdb) ni
0x00005555555555ab 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   0x555555555baf <strings_not_equal>
   0x00005555555555bb <+20>:    test   %eax,%eax
   0x00005555555555bd <+22>:    jne    0x5555555555c4 <phase_1+29>
   0x00005555555555bf <+24>:    add    $0x8,%rsp
   0x00005555555555c3 <+28>:    ret
   0x00005555555555c4 <+29>:    call   0x555555555cc3 <explode_bomb>
```

First putting break point for phase_1 and running the program and then disassemble running it step by step. Then in second line we are seeing wheather the input string is same as 0x555555557150 and then converting this to string using x/s.

```
(gdb) x/s 0x555555557150
0x555555557150: "Border relations with Canada have never been better."
```

Phase_1 answer : Border relations with Canada have never been better.

Phase 2:

```
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 2/PracticeBomb1/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!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
gfsds

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   0x555555555cef <read_six_numbers>
   0x00005555555555ed <+34>:    cmpl   $0x1,(%rsp)
   0x00005555555555f1 <+38>:    jne    0x5555555555fd <phase_2+50>
   0x00005555555555f3 <+40>:    mov    %rsp,%rbx
   0x00005555555555f6 <+43>:    lea    0x14(%rsp),%rbp
   0x00005555555555fb <+48>:    jmp    0x555555555612 <phase_2+71>
   0x00005555555555fd <+50>:    call   0x555555555cc3 <explode_bomb>
   0x0000555555555602 <+55>:    jmp    0x5555555555f3 <phase_2+40>
   0x0000555555555604 <+57>:    call   0x555555555cc3 <explode_bomb>
   0x0000555555555609 <+62>:    add    $0x4,%rbx
   0x000055555555560d <+66>:    cmp    %rbp,%rbx
   0x0000555555555610 <+69>:    je     0x55555555561d <phase_2+82>
   0x0000555555555612 <+71>:    mov    (%rbx),%eax
   0x0000555555555614 <+73>:    add    %eax,%eax
   0x0000555555555616 <+75>:    cmp    %eax,0x4(%rbx)
   0x0000555555555619 <+78>:    je     0x555555555609 <phase_2+62>
```

Putting break point for phase_2 and then running the program. When we disassemble it we get the hint from line6 that in answer there are 6 numbers by function name (read_six_numbers). Then when we are putting stepi and then disas after 7 times we get into read_six_numbers

```
   0x000055555555561b <+80>:    jmp    0x555555555604 <phase_2+57>
   0x000055555555561d <+82>:    mov    0x18(%rsp),%rax
   0x0000555555555622 <+87>:    xor    %fs:0x28,%rax
   0x000055555555562b <+96>:    jne    0x555555555634 <phase_2+105>
   0x000055555555562d <+98>:    add    $0x28,%rsp
   0x0000555555555631 <+102>:   pop    %rbx
   0x0000555555555632 <+103>:   pop    %rbp
   0x0000555555555633 <+104>:   ret
   0x0000555555555634 <+105>:   call   0x555555555220 <__stack_chk_fail@plt>
End of assembler dump.
(gdb) stepi
0x00005555555555cef in read_six_numbers ()
(gdb) disas
Dump of assembler code for function read_six_numbers:
=> 0x0000555555555cef <+0>:     endbr64
   0x0000555555555cf3 <+4>:     sub    $0x8,%rsp
   0x0000555555555cf7 <+8>:     mov    %rsi,%rdx
   0x0000555555555cfa <+11>:    lea    0x4(%rsi),%rcx
   0x0000555555555cfe <+15>:    lea    0x14(%rsi),%rax
   0x0000555555555d02 <+19>:    push   %rax
   0x0000555555555d03 <+20>:    lea    0x10(%rsi),%rax
   0x0000555555555d07 <+24>:    push   %rax
   0x0000555555555d08 <+25>:    lea    0xc(%rsi),%r9
   0x0000555555555d0c <+29>:    lea    0x8(%rsi),%r8
   0x0000555555555d10 <+33>:    lea    0x160c(%rip),%rsi        # 0x555555557323
   0x0000555555555d17 <+40>:    mov    $0x0,%eax
   0x0000555555555d1c <+45>:    call   0x5555555552c0 <__isoc99_sscanf@plt>
   0x0000555555555d21 <+50>:    add    $0x10,%rsp
   0x0000555555555d25 <+54>:    cmp    $0x5,%eax
   0x0000555555555d28 <+57>:    jle    0x555555555d2f <read_six_numbers+64>
   0x0000555555555d2a <+59>:    add    $0x8,%rsp
   0x0000555555555d2e <+63>:    ret
   0x0000555555555d2f <+64>:    call   0x555555555cc3 <explode_bomb>
End of assembler dump.
(gdb) stepi
0x0000555555555cf3 in read_six_numbers ()
(gdb)
0x0000555555555cf7 in read_six_numbers ()
(gdb) disas
Dump of assembler code for function read_six_numbers:
   0x0000555555555cef <+0>:     endbr64
```

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

1) eax = 1
2) eax = eax+eax = 1+1 = 2
3) eax = eax+eax = 2+2 = 4
4) eax = eax+eax = 4+4 = 8
5) eax = eax+eax = 8+8 = 16
6) eax = eax+eax = 16+16 = 32

phase 2 answer : 1 2 4 8 16 32