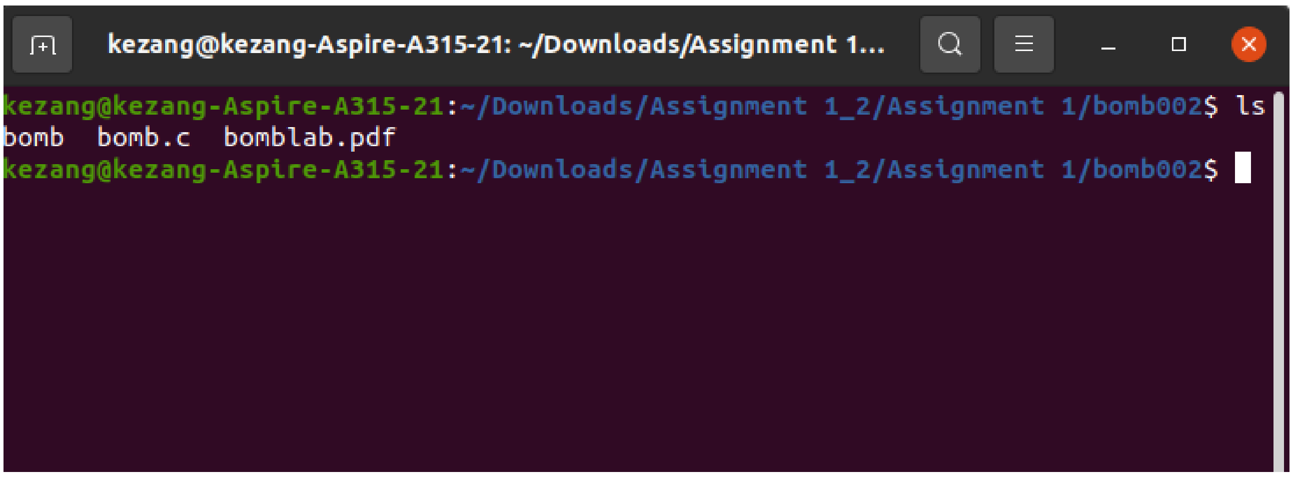
Binary bomb (Group 2)
Phase_1

First phsase_1 is all about giving Dr.evil the correct string and if string is match then bomb is defused.

This are the follow process that I have done in the phase_1 of group 2.

Step 1:

First I have download the zip file from vle and I have exacted here in the download file. After the exaction I have open the group 2 folder and open a terminal.

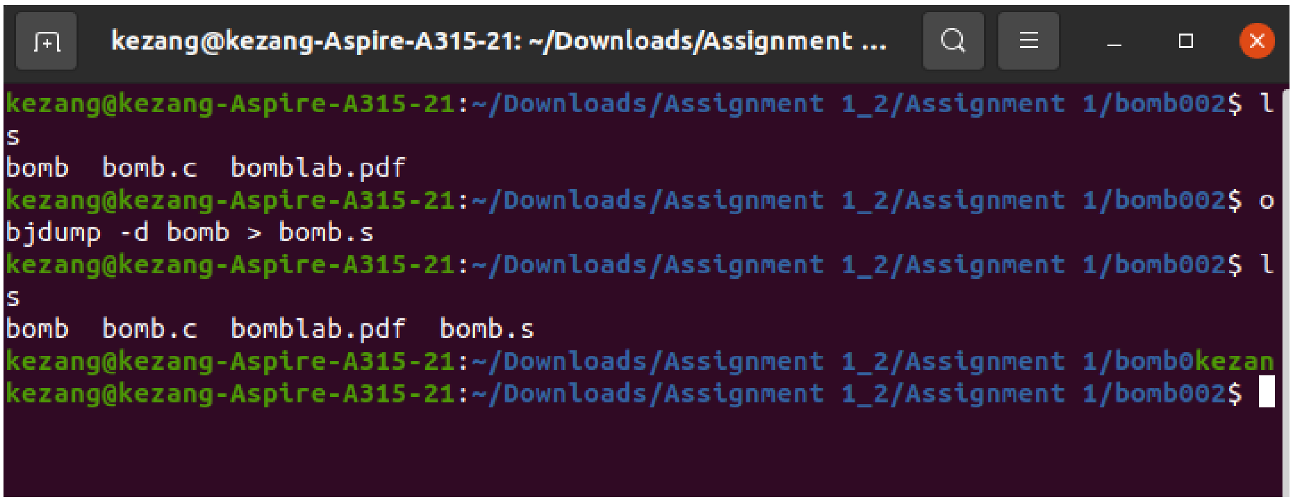


kezang@kezang-Aspire-A315-21: ~/Downloads/Assignment 1...

```
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ ls
bomb   bomb.c   bomblab.pdf
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ 
```

Step 2:

I have used **objdump -d bomb > bomb.s** comand because it will create bomb.s file and disseambler all the code and main functions.
objdump -d command will open the assembler file.
.s is a assembler which mean code are all in assembler form.

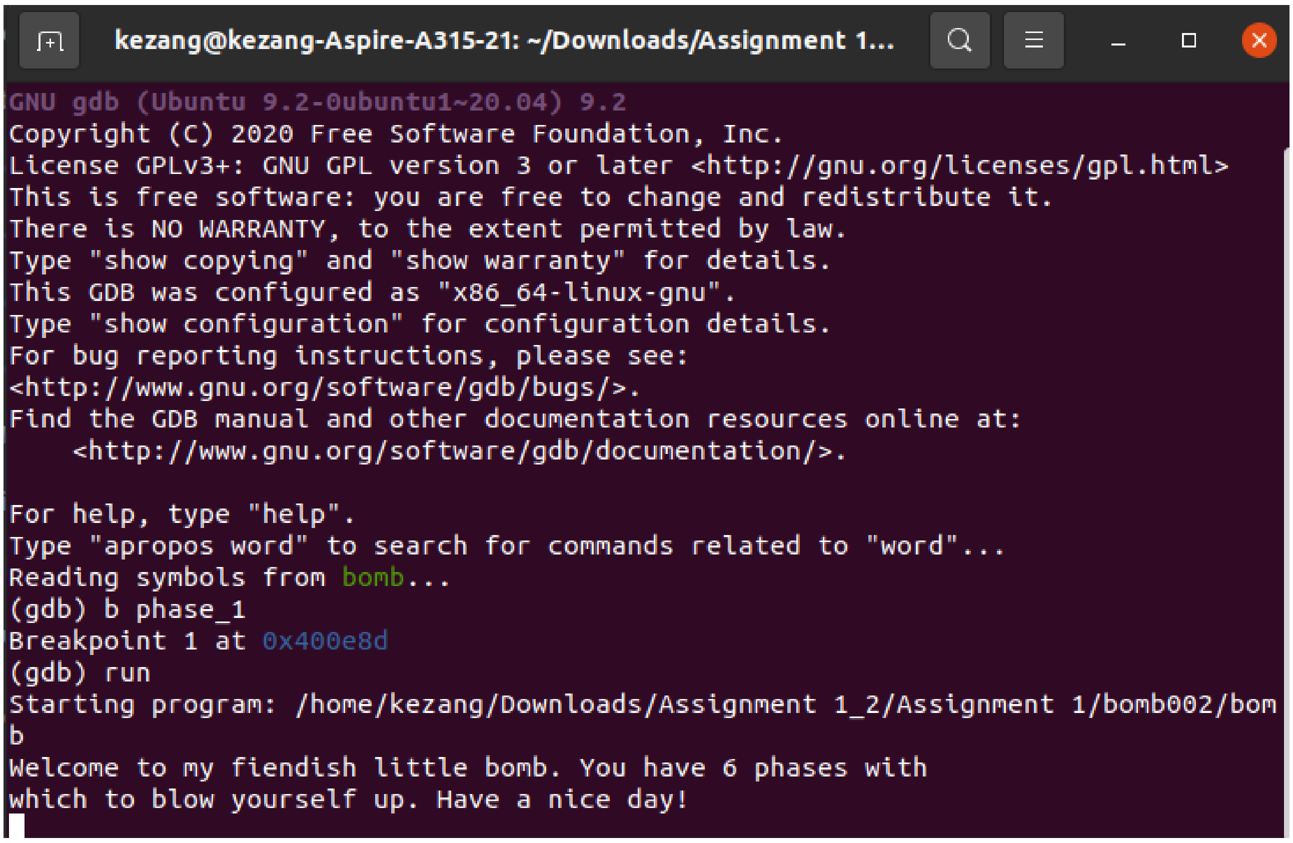


kezang@kezang-Aspire-A315-21: ~/Downloads/Assignment ...

```
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ l
s
bomb   bomb.c   bomblab.pdf
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ o
bjdump -d bomb > bomb.s
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ l
s
bomb   bomb.c   bomblab.pdf   bomb.s
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb0kezan
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ 
```

Step 3:

we have to used **gdb bomb** command in order to open debuger file of c.
gdb help us to run our program up to certain point and print out the value.
After we open we have to setup b point in order to stop phase_1 from debugging process using **b phase_1** command and we **run** the program.

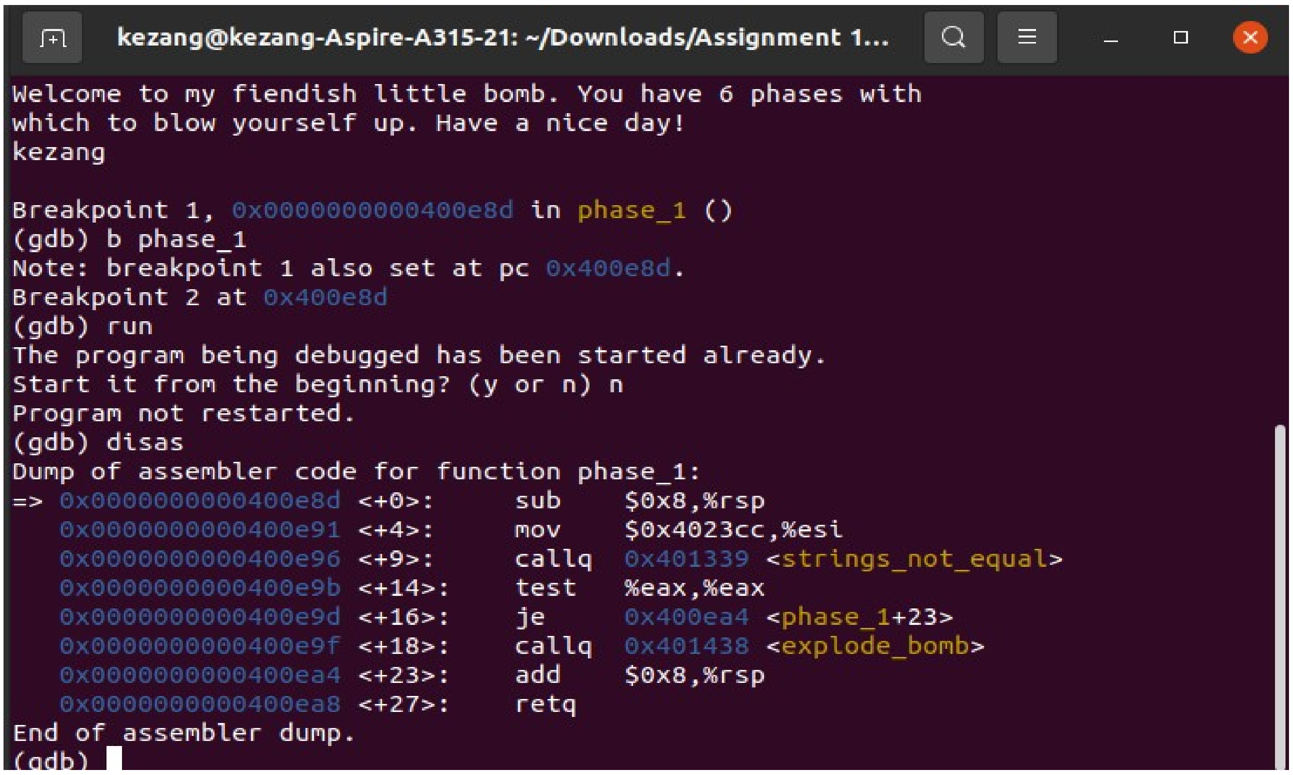


```
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) run
Starting program: /home/kezang/Downloads/Assignment 1_2/Assignment 1/bomb002/bom
b
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```

Step 4:

we have to give any test string and press enter. It will not bomb blast because we have set break point. In this I have given my name as input. We have to used **disas** command in order to go inside the assembler code.



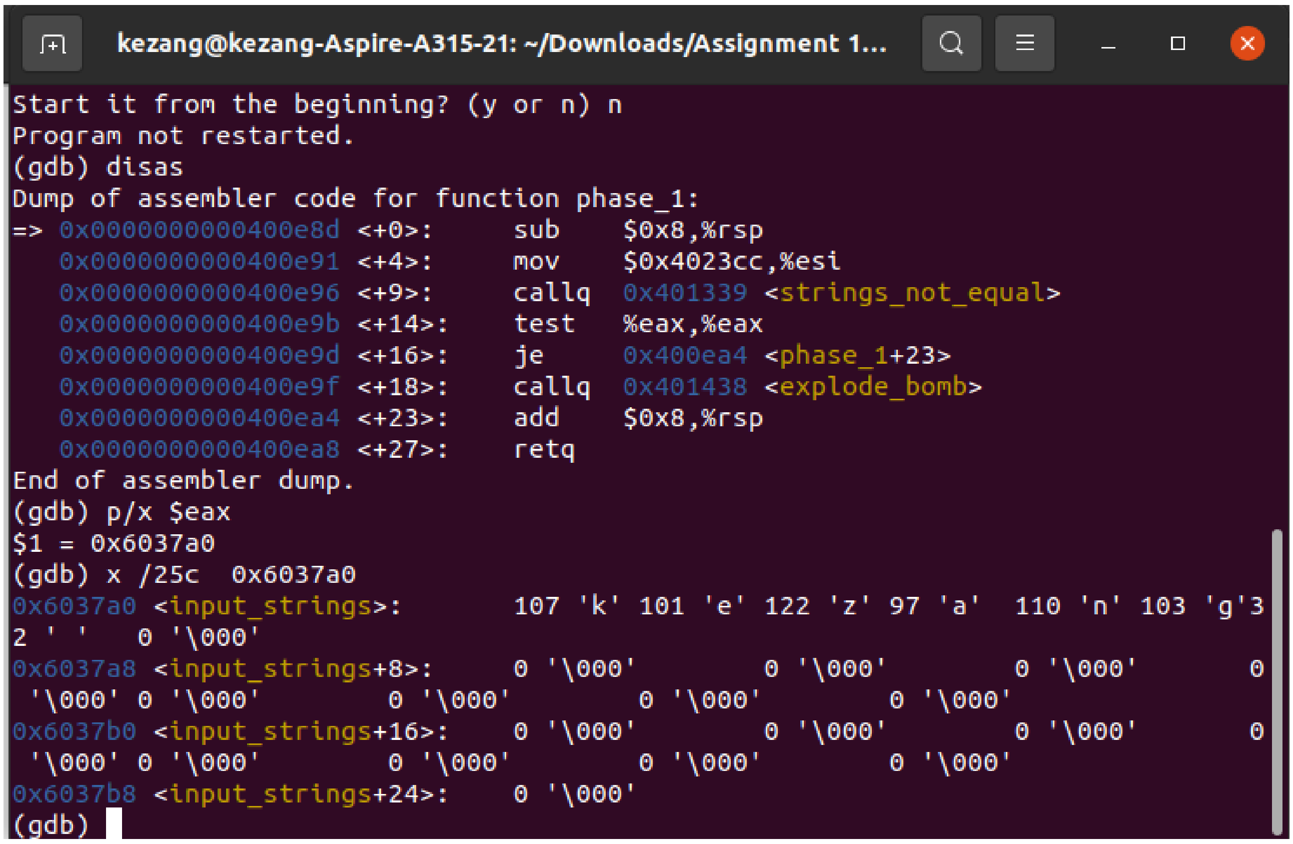
```
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
kezang

Breakpoint 1, 0x0000000000400e8d in phase_1 ()
(gdb) b phase_1
Note: breakpoint 1 also set at pc 0x400e8d.
Breakpoint 2 at 0x400e8d
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x0000000000400e8d <+0>:     sub    $0x8,%rsp
   0x0000000000400e91 <+4>:     mov    $0x4023cc,%esi
   0x0000000000400e96 <+9>:     callq  0x401339 <strings_not_equal>
   0x0000000000400e9b <+14>:    test   %eax,%eax
   0x0000000000400e9d <+16>:    je     0x400ea4 <phase_1+23>
   0x0000000000400e9f <+18>:    callq  0x401438 <explode_bomb>
   0x0000000000400ea4 <+23>:    add    $0x8,%rsp
   0x0000000000400ea8 <+27>:    retq
End of assembler dump.
(gdb)
```

After entering disas command we get dump assembler code for function phase_1. In this we have callq function which calls <strings_not_equal> we gives us hint that our input is in string not a interger. In line <+14> we have %eax and %eax register in which it compare user input and input set by Dr.evil if it is match then it will jump to <phase_1+23> line if not then then bomb will explode.

Step 5:

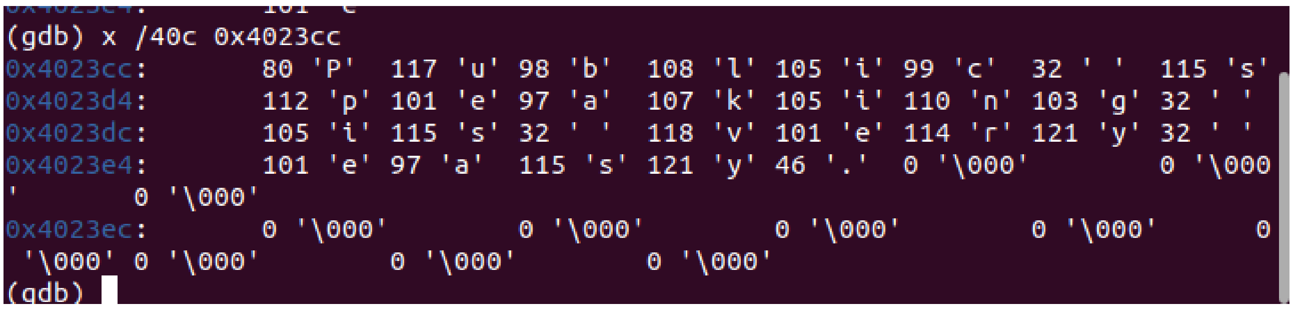
we use **p/x $eax** command in order to get address of eax register and address we got is 0x6037a0.
After that we used **x /25c 0x6037a0** command in order to get test string that I have put early.



```
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x0000000000400e8d <+0>:     sub    $0x8,%rsp
   0x0000000000400e91 <+4>:     mov    $0x4023cc,%esi
   0x0000000000400e96 <+9>:     callq  0x401339 <strings_not_equal>
   0x0000000000400e9b <+14>:    test   %eax,%eax
   0x0000000000400e9d <+16>:    je     0x400ea4 <phase_1+23>
   0x0000000000400e9f <+18>:    callq  0x401438 <explode_bomb>
   0x0000000000400ea4 <+23>:    add    $0x8,%rsp
   0x0000000000400ea8 <+27>:    retq
End of assembler dump.
(gdb) p/x $eax
$1 = 0x6037a0
(gdb) x /25c  0x6037a0
0x6037a0 <input_strings>:       107 'k' 101 'e' 122 'z' 97 'a'  110 'n' 103 'g'3
2 ' '   0 '\000'
0x6037a8 <input_strings+8>:     0 '\000'        0 '\000'        0 '\000'        0
 '\000' 0 '\000'        0 '\000'        0 '\000'        0 '\000'
0x6037b0 <input_strings+16>:    0 '\000'        0 '\000'        0 '\000'        0
 '\000' 0 '\000'        0 '\000'        0 '\000'        0 '\000'
0x6037b8 <input_strings+24>:    0 '\000'
(gdb)
```

Step 6:

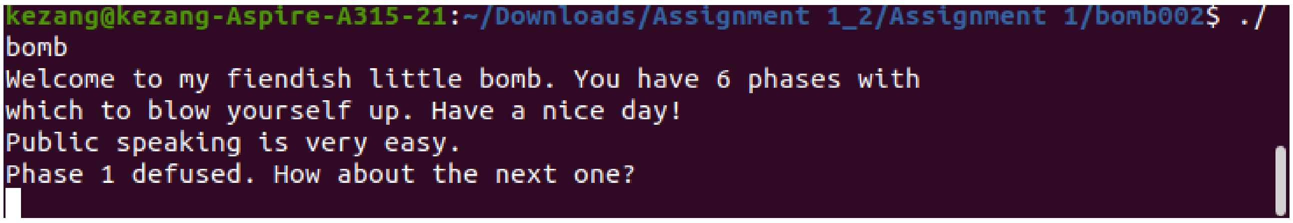
In order to get the string of Dr.evil we have to go to <+4> line since the esi has been move to 0x4023cc. So we have to put **x /40c 0x4023cc** and we get the string that Dr.evil have set in order to defuse the bomb.



```
(gdb) x /40c 0x4023cc
0x4023cc:       80 'P'  117 'u' 98 'b'  108 'l' 105 'i' 99 'c'  32 ' '  115 's'
0x4023d4:       112 'p' 101 'e' 97 'a'  107 'k' 105 'i' 110 'n' 103 'g' 32 ' '
0x4023dc:       105 'i' 115 's' 32 ' '  118 'v' 101 'e' 114 'r' 121 'y' 32 ' '
0x4023e4:       101 'e' 97 'a'  115 's' 121 'y' 46 '.'  0 '\000'        0 '\000
'       0 '\000'
0x4023ec:       0 '\000'        0 '\000'        0 '\000'        0 '\000'        0
 '\000' 0 '\000'        0 '\000'        0 '\000'
(gdb)
```

Step 7:

From the above process we get the answer set by Dr.evil which is (**public speaking is very easy.).** After that we have to close gdb file and we have to enter **./bomb** command and enter the string that we got.



```
kezang@kezang-Aspire-A315-21:~/Downloads/Assignment 1_2/Assignment 1/bomb002$ ./
bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
```

After entering the string we have successfully defuse a bomb and we can proceed to next phase.