ASSIGNMENT-3 SE LAB

NAME: SAMUDRA ROY ROLL:002211001114 SE:A3

1. Consider the program in Assign3.It is a simple state machine.

```
"e.c" 61L, 1209C written
[be22114@localhost ~]$ gcc -g e.c -o e
[be22114@localhost ~]$ gdb e
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-94.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/usr/student/ug/yr22/be22114/e...done.
(gdb)
```

a. Put a breakpoint in line 49

ANS: break 49

```
(gdb) break 49
Breakpoint 1 at 0x4006a1: file e.c, line 49.
(gdb)
```

b. Try next command

```
Breakpoint 1, main () at e.c:49
49              step_state(events_arr[cntr]);
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) next
50              cntr++;
(gdb)
```

c. How will you get inside the function without

using breakpoint?

Ans: step=> Runs the next instruction, not line. If the current instruction is setting a variable, it is the same as next. If it's a function, it will jump into the function, execute the first statement, then pause.

```
Breakpoint 1, main () at e.c:49
49              step_state(events_arr[cntr]);
(gdb) step
step_state (event=START_LOOPING) at e.c:15
15          switch(state) {
(gdb) step
17              switch(event) {
(gdb) step
19                  state = LOOP;
(gdb)
```

d. How will you come out the of the function

without using next and continue?

Ans: finish => Finishes executing the current function, then pause (also called step out).

```
(gdb) finish
Run till exit from #0  step_state (event=START_LOOPING) at e.c:19
main () at e.c:50
50              cntr++;
(gdb) step
47          while(events_arr[cntr] != STOP_LOOPING)
(gdb)
```

2. Consider the program in Assign4 .It is also a simple

state machine.If you provide user id and password properly account details will be displayed. The basic rule is user id should be positive and less than 20 .password is userid *b1000 .The loop will terminate after 10 iteration. It works fine if you provide valid user id and password.It works fine for invalid userid. But it goes to infiniteloop for invalid password.Run the program .It goes into infinite loop.you need to kill the program by [ctrl^c]

```
"f.c" [New] 106L, 1857C written
[be22114@localhost ~]$ gcc -g f.c -o f
[be22114@localhost ~]$ gdb f
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-94.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/usr/student/ug/yr22/be22114/f...done.
(gdb)
```

a. Set a suitable breakpoint in gdb in the routine show.give valid input and run :

break 43

```
(gdb) b 43
Breakpoint 1 at 0x4006bb: file f.c, line 43.
(gdb) ^CQuit
(gdb) ^CQuit
(gdb)
Note: breakpoint 1 also set at pc 0x4006bb.
Breakpoint 2 at 0x4006bb: file f.c, line 43.
(gdb) ^CQuit
(gdb) q
[be22114@localhost ~]$ ls
a                 assignment1.c    assignment2,6.c  assignment3,3.c  assignment4,3.c  ass
a.c               assignment2,1.c  assignment2,7.c  assignment3,4.c  assignment4,4.c  ass
a.cpp             assignment2,2.c  assignment2,8.c  assignment3,6.c  assignment4.c    ass
addtwonumber.cpp  assignment2,3.c  assignment2.c    assignment3.c    assignment5,1.c  ass
a.h               assignment2,4.c  assignment3,1.c  assignment4,1.c  assignment5,2.c  ass
a.out             assignment2,5.c  assignment3,2.c  assignment4,2.c  assignment5,3.c  ass
[be22114@localhost ~]$ gcc -g f.c -o f
[be22114@localhost ~]$ gdb f
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-94.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/usr/student/ug/yr22/be22114/f...done.
(gdb) b show
Breakpoint 1 at 0x40067b: file f.c, line 35.
(gdb)
```

b. How you can see the call stack of the routine.

Ans: bt

```
Breakpoint 1 at 0x40067b: file f.c, line 35.
(gdb) r
Starting program: /home/usr/student/ug/yr22/be22114/f
Hello Please Provide  User Id and Password to see your details!
User Id: 4
Password: 4000

Breakpoint 1, show (id=4) at f.c:35
35          return id*100000;
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) bt
#0  show (id=4) at f.c:35
#1  0x0000000000400805 in step_state (event=SHOW_DETAIL) at f.c:93
#2  0x000000000040085a in main () at f.c:111
(gdb)
```

c. Which commands will help you to see each value change of variable “event”?

Ans: Watch event

```
Reading symbols from /home/usr/student/ug/yr22/be22114/f...done.
(gdb) b 43
Breakpoint 1 at 0x4006bb: file f.c, line 43.
(gdb) r
Starting program: /home/usr/student/ug/yr22/be22114/f

Breakpoint 1, step_state (event=START_LOOPING) at f.c:44
44              switch(event) {
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) c
Continuing.
Hello Please Provide  User Id and Password to see your details!
User Id: 4
Password: 4000
User Id : 4, Password: 4000 , Amount : 400000

Breakpoint 1, step_state (event=START_LOOPING) at f.c:44
44              switch(event) {
(gdb) watch event
Hardware watchpoint 2: event
(gdb) c
Continuing.
Hello Please Provide  User Id and Password to see your details!
User Id: 4
Hardware watchpoint 2: event

Old value = START_LOOPING
New value = USERID_MATCHED
0x0000000000400746 in step_state (event=USERID_MATCHED) at f.c:57
57                  event = USERID_MATCHED ;
(gdb) c
Continuing.
Password: 4000
Hardware watchpoint 2: event

Old value = USERID_MATCHED
New value = SHOW_DETAIL
```

```
step_state (event=SHOW_DETAIL) at f.c:89
89                  break;
(gdb) c
Continuing.
User Id : 4, Password: 4000 , Amount : 400000
Hardware watchpoint 2: event

Old value = SHOW_DETAIL
New value = START_LOOPING
step_state (event=START_LOOPING) at f.c:98
98                  break;
(gdb)
```

d. Correct the program so that it doesn't go to infinite loop for wrong password. Rather main iteration restarts . [follow the value change path of event for wrong password]

ans:

Changed *event = STOP_LOOPING* to *event=START_LOOPING* at line 94/95

```
    case LOOP:
        switch(event) {
        case USERID_MATCHED:
   printf("Password: ");
   scanf("%d", &password);
   if (valid_pw(id,password)) {
   event = SHOW_DETAIL ;
   } else {
   printf("Incorrect password!!\n");
   event = STOP_LOOPING ;
   state = START ;
   }
            break;
        case SHOW_DETAIL:
  {
  char c = 'p';
   printf("User Id : %d, Password: %d , Amount : %d\n", id,password,show(id));
   state = START ;
   event = START_LOOPING;

   }
            break;
        default:
            exit(1);
            break;
        }
        break;
```

Explore the commands found for 5c to see/use content of a pointer

```
Breakpoint 1, step_state (event=USERID_MATCHED) at f.c:94
94                          printf("Incorrect password!!\n");
Missing separate debuginfos, use: debuginfo-install glibc-2.17-15
7.el7_3.2.x86_64
(gdb) watch event
Hardware watchpoint 2: event
(gdb) n
Incorrect password!!
95                          event = START_LOOPING ;
(gdb) n
Hardware watchpoint 2: event

Old value = USERID_MATCHED
New value = START_LOOPING
step_state (event=START_LOOPING) at f.c:96
96                          state =  START;
```