Module 4 Assessment Solutions

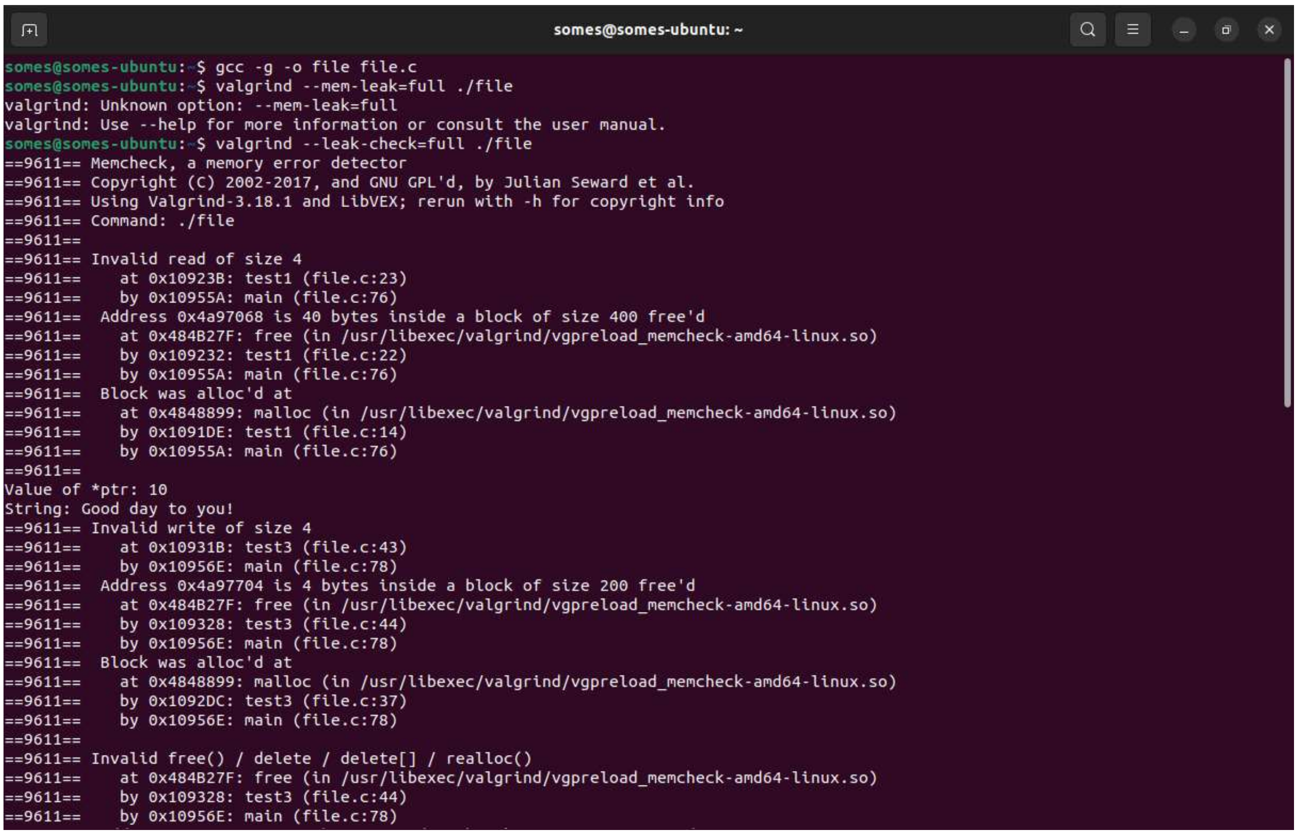
1) Using Valgind identify memleaks in the given program. Explore optional flags in Valgrind.

Identifying the memleaks in the given program.

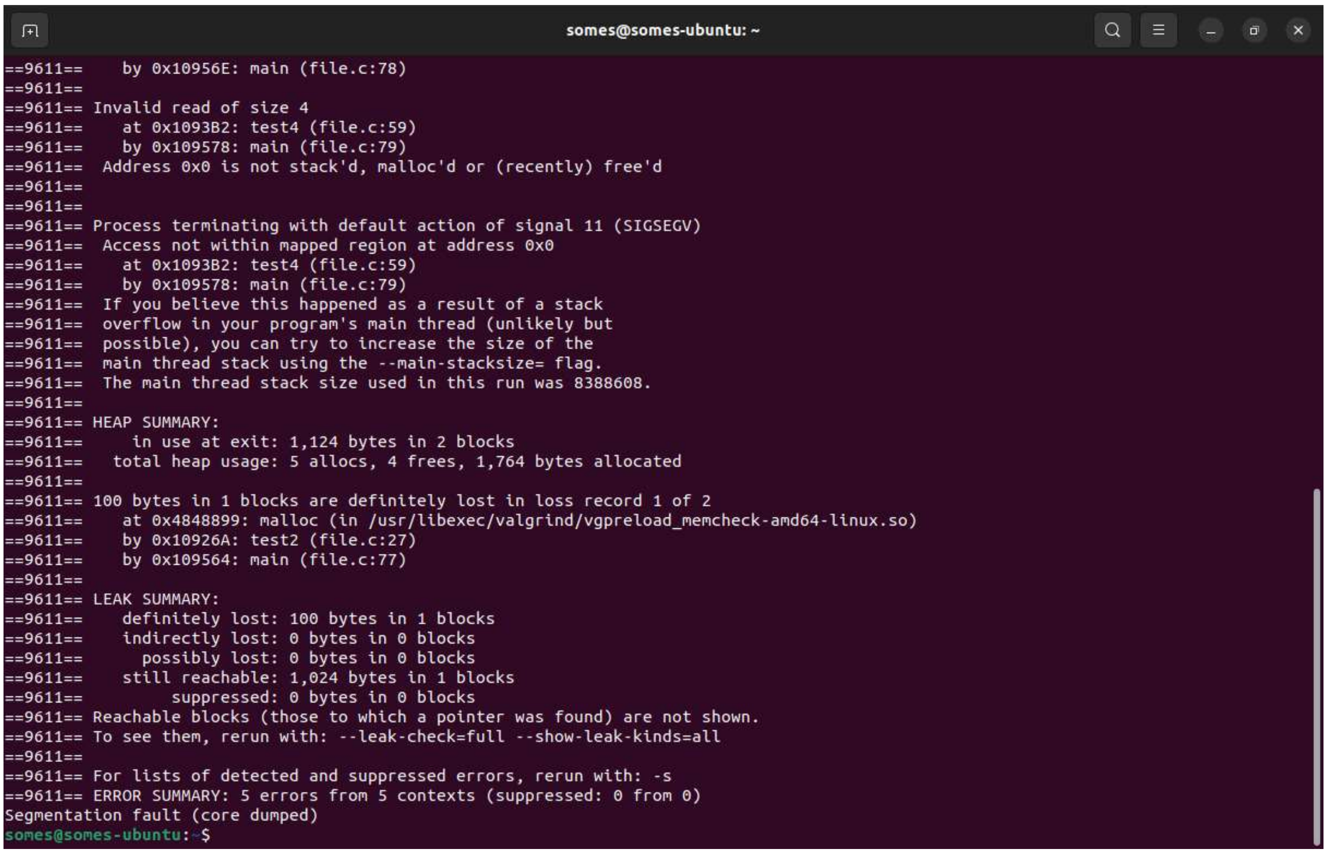
First we must add the debugging files into the code : ***gcc -g -o file file.c***

Here “-g” adds the debugging information and “-o file” creates the output file.

Once the debugging info is added to the file, we must run the valgrind in order to check for memleaks : valgrind –“***mem-check=full ./file”***



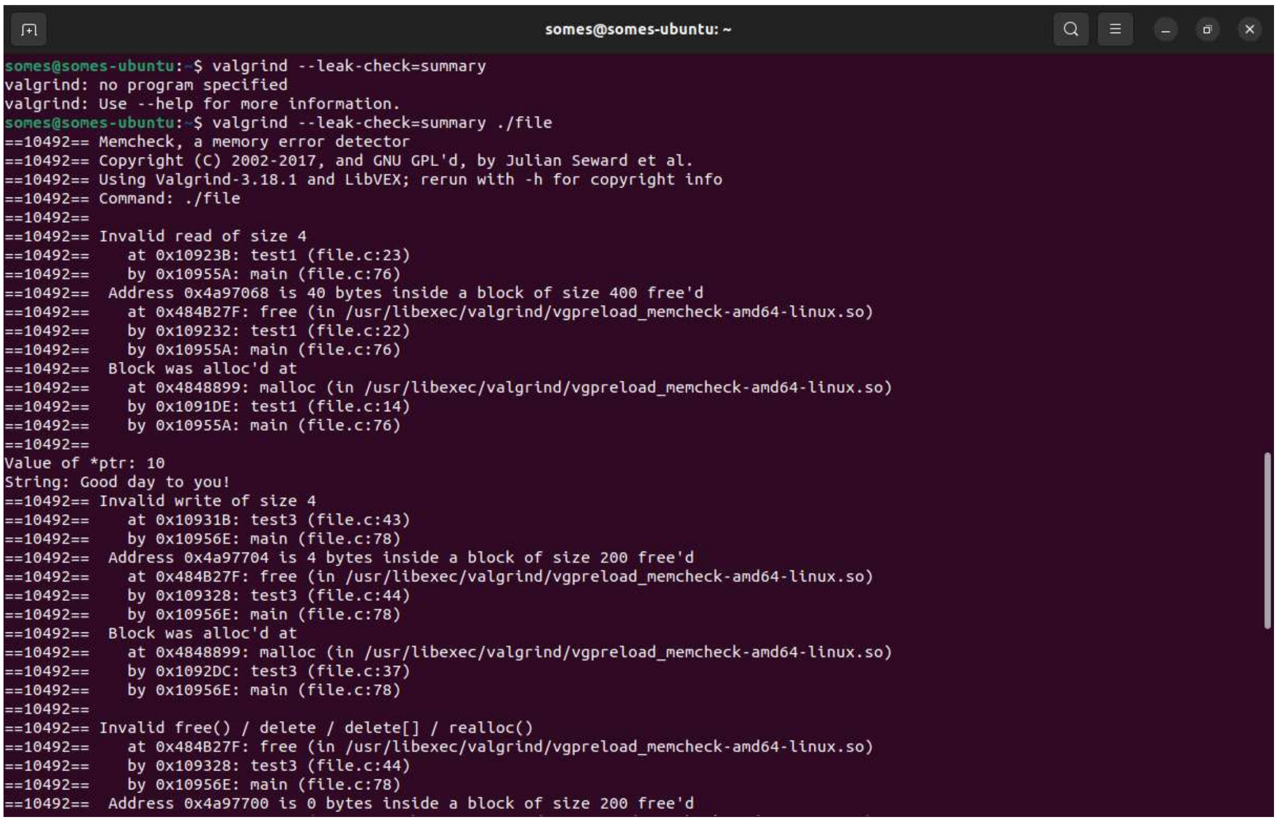
```
somes@somes-ubuntu:~$ gcc -g -o file file.c
somes@somes-ubuntu:~$ valgrind --mem-leak=full ./file
valgrind: Unknown option: --mem-leak=full
valgrind: Use --help for more information or consult the user manual.
somes@somes-ubuntu:~$ valgrind --leak-check=full ./file
==9611== Memcheck, a memory error detector
==9611== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9611== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==9611== Command: ./file
==9611==
==9611== Invalid read of size 4
==9611==    at 0x10923B: test1 (file.c:23)
==9611==    by 0x10955A: main (file.c:76)
==9611==  Address 0x4a97068 is 40 bytes inside a block of size 400 free'd
==9611==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==9611==    by 0x109232: test1 (file.c:22)
==9611==    by 0x10955A: main (file.c:76)
==9611==  Block was alloc'd at
==9611==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==9611==    by 0x1091DE: test1 (file.c:14)
==9611==    by 0x10955A: main (file.c:76)
==9611==
Value of *ptr: 10
String: Good day to you!
==9611== Invalid write of size 4
==9611==    at 0x10931B: test3 (file.c:43)
==9611==    by 0x10956E: main (file.c:78)
==9611==  Address 0x4a97704 is 4 bytes inside a block of size 200 free'd
==9611==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==9611==    by 0x109328: test3 (file.c:44)
==9611==    by 0x10956E: main (file.c:78)
==9611==  Block was alloc'd at
==9611==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==9611==    by 0x1092DC: test3 (file.c:37)
==9611==    by 0x10956E: main (file.c:78)
==9611==
==9611== Invalid free() / delete / delete[] / realloc()
==9611==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==9611==    by 0x109328: test3 (file.c:44)
==9611==    by 0x10956E: main (file.c:78)
```



```
==9611==    by 0x10956E: main (file.c:78)
==9611==
==9611== Invalid read of size 4
==9611==    at 0x1093B2: test4 (file.c:59)
==9611==    by 0x109578: main (file.c:79)
==9611==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==9611==
==9611==
==9611== Process terminating with default action of signal 11 (SIGSEGV)
==9611==  Access not within mapped region at address 0x0
==9611==    at 0x1093B2: test4 (file.c:59)
==9611==    by 0x109578: main (file.c:79)
==9611==  If you believe this happened as a result of a stack
==9611==  overflow in your program's main thread (unlikely but
==9611==  possible), you can try to increase the size of the
==9611==  main thread stack using the --main-stacksize= flag.
==9611==  The main thread stack size used in this run was 8388608.
==9611==
==9611== HEAP SUMMARY:
==9611==     in use at exit: 1,124 bytes in 2 blocks
==9611==   total heap usage: 5 allocs, 4 frees, 1,764 bytes allocated
==9611==
==9611== 100 bytes in 1 blocks are definitely lost in loss record 1 of 2
==9611==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==9611==    by 0x10926A: test2 (file.c:27)
==9611==    by 0x109564: main (file.c:77)
==9611==
==9611== LEAK SUMMARY:
==9611==    definitely lost: 100 bytes in 1 blocks
==9611==    indirectly lost: 0 bytes in 0 blocks
==9611==      possibly lost: 0 bytes in 0 blocks
==9611==    still reachable: 1,024 bytes in 1 blocks
==9611==         suppressed: 0 bytes in 0 blocks
==9611== Reachable blocks (those to which a pointer was found) are not shown.
==9611== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==9611==
==9611== For lists of detected and suppressed errors, rerun with: -s
==9611== ERROR SUMMARY: 5 errors from 5 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
somes@somes-ubuntu:~$
```

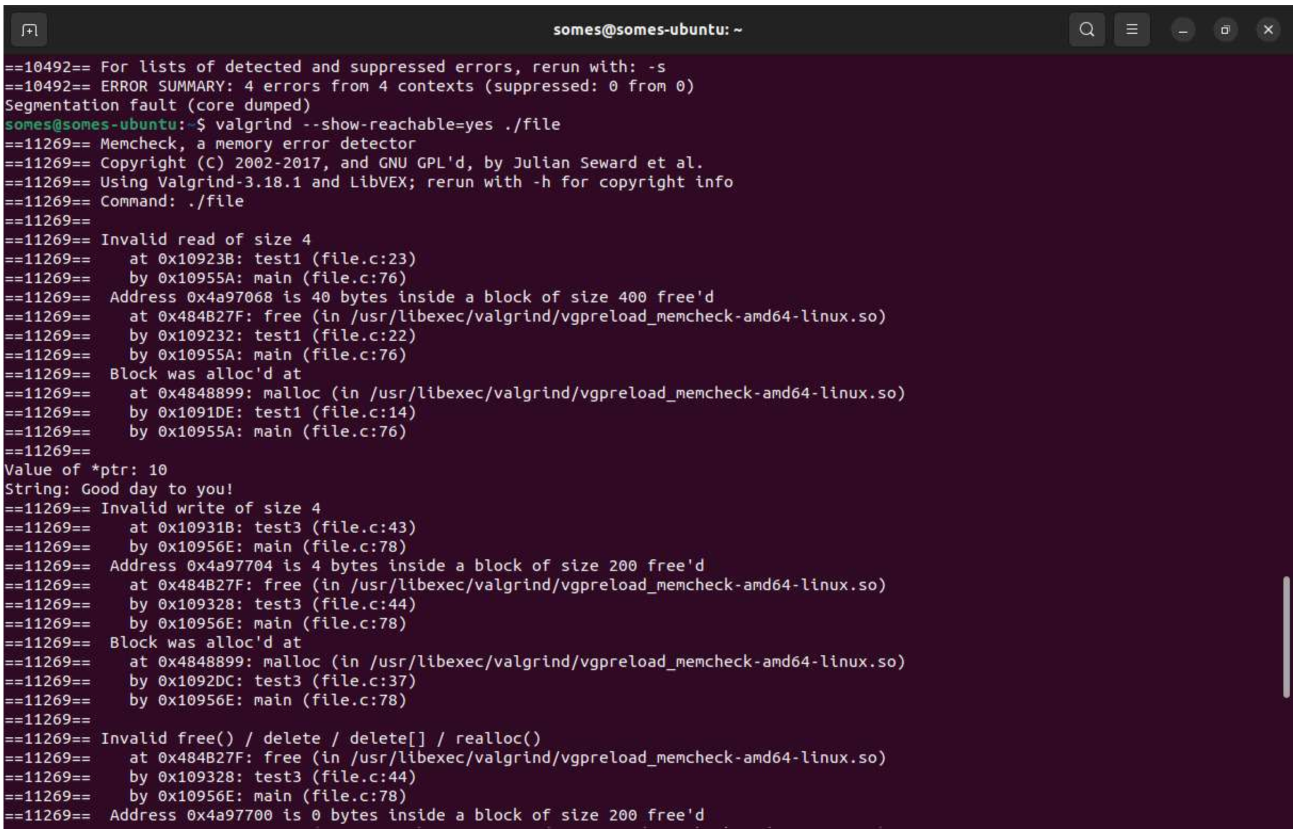
Explore optional flags in valgrind

To check the summary of all the leaks : ***“valgrind –leak-check=summary ./file”***

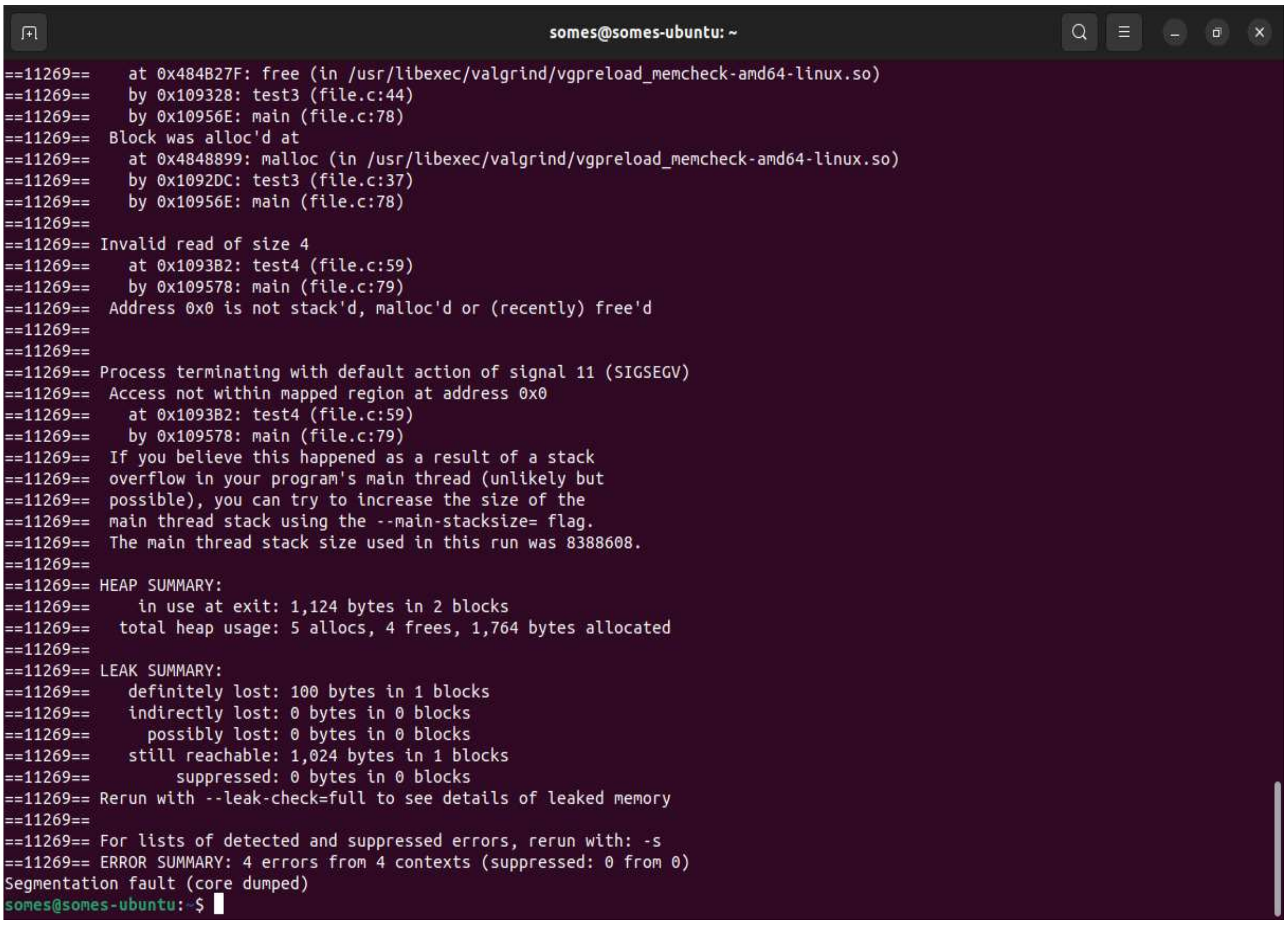


```
somes@somes-ubuntu:~$ valgrind --leak-check=summary
valgrind: no program specified
valgrind: Use --help for more information.
somes@somes-ubuntu:~$ valgrind --leak-check=summary ./file
==10492== Memcheck, a memory error detector
==10492== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10492== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10492== Command: ./file
==10492==
==10492== Invalid read of size 4
==10492==    at 0x10923B: test1 (file.c:23)
==10492==    by 0x10955A: main (file.c:76)
==10492==  Address 0x4a97068 is 40 bytes inside a block of size 400 free'd
==10492==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==10492==    by 0x109232: test1 (file.c:22)
==10492==    by 0x10955A: main (file.c:76)
==10492==  Block was alloc'd at
==10492==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==10492==    by 0x1091DE: test1 (file.c:14)
==10492==    by 0x10955A: main (file.c:76)
==10492==
Value of *ptr: 10
String: Good day to you!
==10492== Invalid write of size 4
==10492==    at 0x10931B: test3 (file.c:43)
==10492==    by 0x10956E: main (file.c:78)
==10492==  Address 0x4a97704 is 4 bytes inside a block of size 200 free'd
==10492==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==10492==    by 0x109328: test3 (file.c:44)
==10492==    by 0x10956E: main (file.c:78)
==10492==  Block was alloc'd at
==10492==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==10492==    by 0x1092DC: test3 (file.c:37)
==10492==    by 0x10956E: main (file.c:78)
==10492==
==10492== Invalid free() / delete / delete[] / realloc()
==10492==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==10492==    by 0x109328: test3 (file.c:44)
==10492==    by 0x10956E: main (file.c:78)
==10492==  Address 0x4a97700 is 0 bytes inside a block of size 200 free'd
```

***Valgrind –show-reachable=yes ./file***



```
==10492== For lists of detected and suppressed errors, rerun with: -s
==10492== ERROR SUMMARY: 4 errors from 4 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
somes@somes-ubuntu:~$ valgrind --show-reachable=yes ./file
==11269== Memcheck, a memory error detector
==11269== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11269== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==11269== Command: ./file
==11269==
==11269== Invalid read of size 4
==11269==    at 0x10923B: test1 (file.c:23)
==11269==    by 0x10955A: main (file.c:76)
==11269==  Address 0x4a97068 is 40 bytes inside a block of size 400 free'd
==11269==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x109232: test1 (file.c:22)
==11269==    by 0x10955A: main (file.c:76)
==11269==  Block was alloc'd at
==11269==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x1091DE: test1 (file.c:14)
==11269==    by 0x10955A: main (file.c:76)
==11269==
Value of *ptr: 10
String: Good day to you!
==11269== Invalid write of size 4
==11269==    at 0x10931B: test3 (file.c:43)
==11269==    by 0x10956E: main (file.c:78)
==11269==  Address 0x4a97704 is 4 bytes inside a block of size 200 free'd
==11269==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x109328: test3 (file.c:44)
==11269==    by 0x10956E: main (file.c:78)
==11269==  Block was alloc'd at
==11269==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x1092DC: test3 (file.c:37)
==11269==    by 0x10956E: main (file.c:78)
==11269==
==11269== Invalid free() / delete / delete[] / realloc()
==11269==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x109328: test3 (file.c:44)
==11269==    by 0x10956E: main (file.c:78)
==11269==  Address 0x4a97700 is 0 bytes inside a block of size 200 free'd
```



```
==11269==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x109328: test3 (file.c:44)
==11269==    by 0x10956E: main (file.c:78)
==11269==  Block was alloc'd at
==11269==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==11269==    by 0x1092DC: test3 (file.c:37)
==11269==    by 0x10956E: main (file.c:78)
==11269==
==11269== Invalid read of size 4
==11269==    at 0x1093B2: test4 (file.c:59)
==11269==    by 0x109578: main (file.c:79)
==11269==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==11269==
==11269==
==11269== Process terminating with default action of signal 11 (SIGSEGV)
==11269==  Access not within mapped region at address 0x0
==11269==    at 0x1093B2: test4 (file.c:59)
==11269==    by 0x109578: main (file.c:79)
==11269==  If you believe this happened as a result of a stack
==11269==  overflow in your program's main thread (unlikely but
==11269==  possible), you can try to increase the size of the
==11269==  main thread stack using the --main-stacksize= flag.
==11269==  The main thread stack size used in this run was 8388608.
==11269==
==11269== HEAP SUMMARY:
==11269==     in use at exit: 1,124 bytes in 2 blocks
==11269==   total heap usage: 5 allocs, 4 frees, 1,764 bytes allocated
==11269==
==11269== LEAK SUMMARY:
==11269==    definitely lost: 100 bytes in 1 blocks
==11269==    indirectly lost: 0 bytes in 0 blocks
==11269==      possibly lost: 0 bytes in 0 blocks
==11269==    still reachable: 1,024 bytes in 1 blocks
==11269==         suppressed: 0 bytes in 0 blocks
==11269== Rerun with --leak-check=full to see details of leaked memory
==11269==
==11269== For lists of detected and suppressed errors, rerun with: -s
==11269== ERROR SUMMARY: 4 errors from 4 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
somes@somes-ubuntu:~$
```

By default, Valgrind reports only "definitely lost" and "indirectly lost" memory blocks as memory leaks. The --show-reachable=yes option makes Valgrind also report "reachable" and "possibly lost" memory blocks.

2) With the same program, using GDB, set breakpoints, run the program, list the code, run from one breakpoint to another, print the value of variables while execution, check assemble code, disable breakpoints, check registers info, explore optional flags.

Setting breakpoints

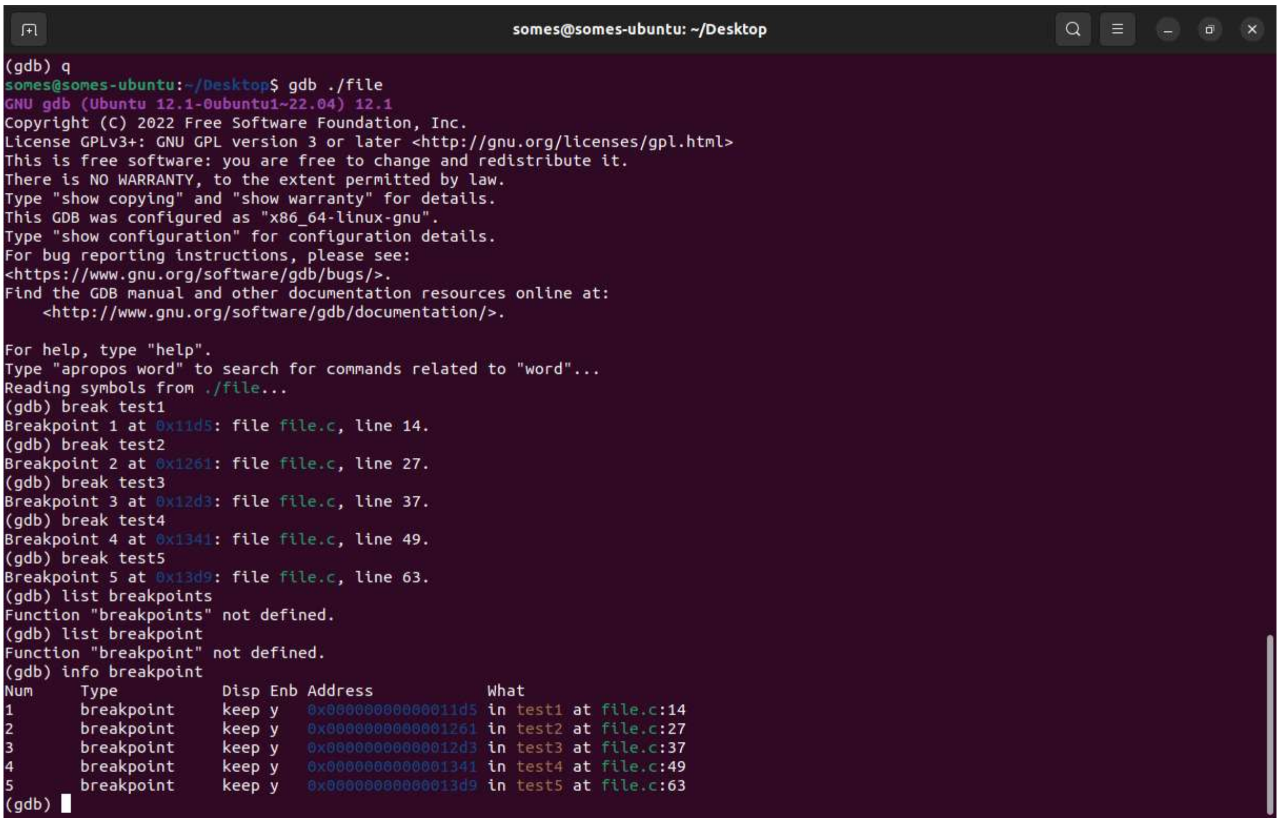
To compile the file with debugging info – "***gcc -g -o file file.c***"

To start gbd on the code – "***gdb ./file***"

"***break main***" – sets breakpoint from main function

"***break test1***" – sets breakpoint at a specific function, here it is "test1"

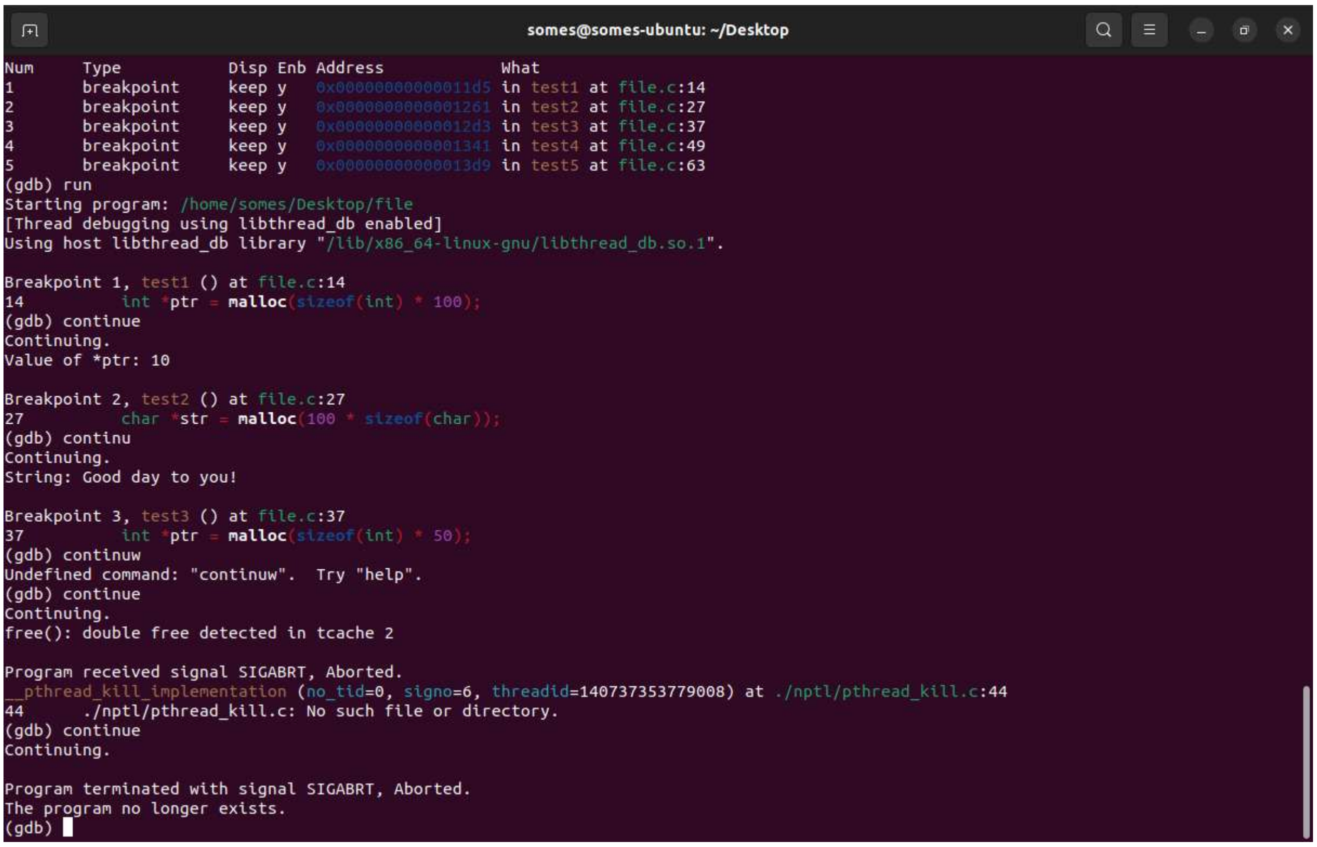
"***break 21***" - sets breakpoint at line number 21.



```
(gdb) q
somes@somes-ubuntu:~/Desktop$ gdb ./file
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 ./file...
(gdb) break test1
Breakpoint 1 at 0x11d5: file file.c, line 14.
(gdb) break test2
Breakpoint 2 at 0x1261: file file.c, line 27.
(gdb) break test3
Breakpoint 3 at 0x12d3: file file.c, line 37.
(gdb) break test4
Breakpoint 4 at 0x1341: file file.c, line 49.
(gdb) break test5
Breakpoint 5 at 0x13d9: file file.c, line 63.
(gdb) list breakpoints
Function "breakpoints" not defined.
(gdb) list breakpoint
Function "breakpoint" not defined.
(gdb) info breakpoint
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011d5 in test1 at file.c:14
2       breakpoint     keep y   0x0000000000001261 in test2 at file.c:27
3       breakpoint     keep y   0x00000000000012d3 in test3 at file.c:37
4       breakpoint     keep y   0x0000000000001341 in test4 at file.c:49
5       breakpoint     keep y   0x00000000000013d9 in test5 at file.c:63
(gdb) 
```

After setting breakpoints, we must run the code

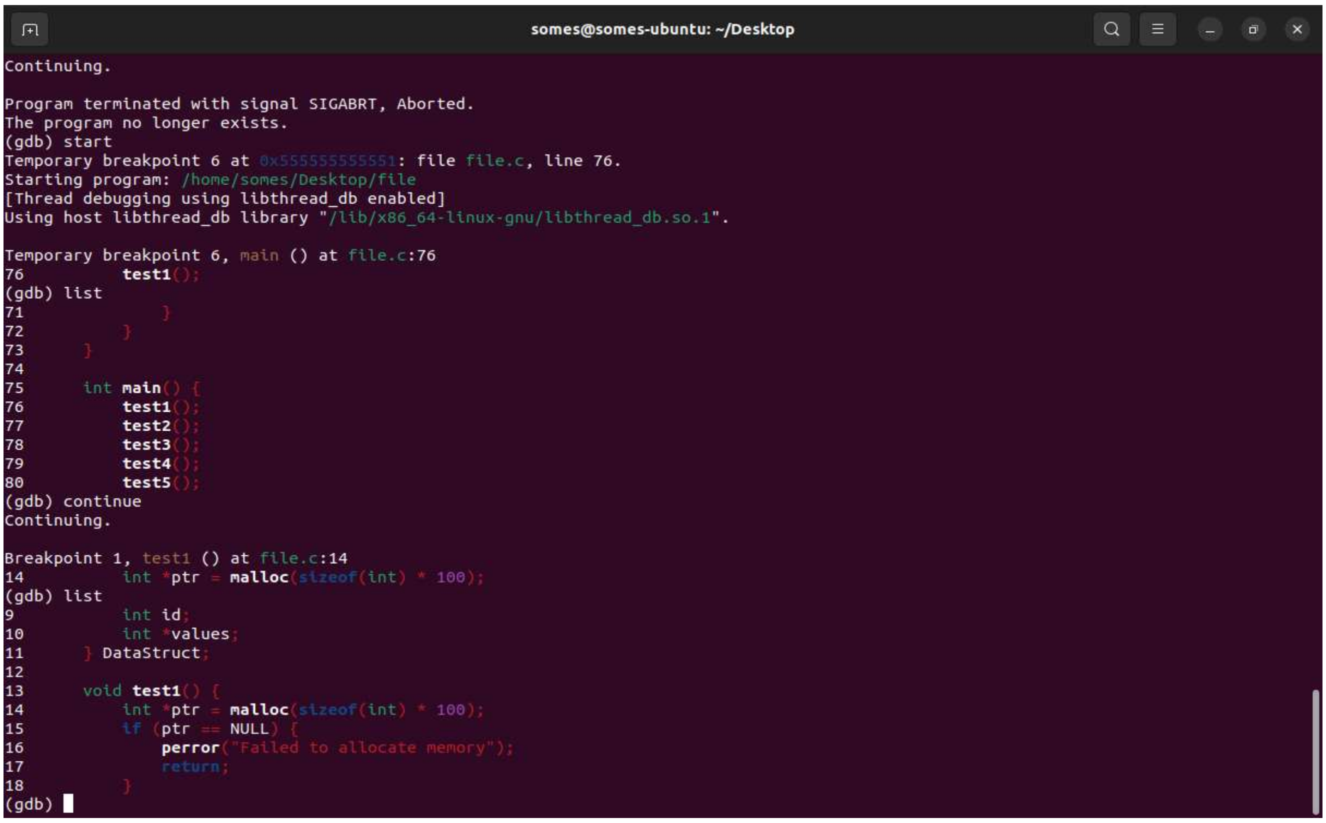
(***gdb) run***



```
somes@somes-ubuntu: ~/Desktop
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011d5 in test1 at file.c:14
2       breakpoint     keep y   0x0000000000001261 in test2 at file.c:27
3       breakpoint     keep y   0x00000000000012d3 in test3 at file.c:37
4       breakpoint     keep y   0x0000000000001341 in test4 at file.c:49
5       breakpoint     keep y   0x00000000000013d9 in test5 at file.c:63
(gdb) run
Starting program: /home/somes/Desktop/file
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, test1 () at file.c:14
14          int *ptr = malloc(sizeof(int) * 100);
(gdb) continue
Continuing.
Value of *ptr: 10

Breakpoint 2, test2 () at file.c:27
27          char *str = malloc(100 * sizeof(char));
(gdb) continu
Continuing.
String: Good day to you!

Breakpoint 3, test3 () at file.c:37
37          int *ptr = malloc(sizeof(int) * 50);
(gdb) continuw
Undefined command: "continuw".  Try "help".
(gdb) continue
Continuing.
free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (no_tid=0, signo=6, threadid=140737353779008) at ./nptl/pthread_kill.c:44
44      ./nptl/pthread_kill.c: No such file or directory.
(gdb) continue
Continuing.

Program terminated with signal SIGABRT, Aborted.
The program no longer exists.
(gdb) 
```

Listing the code

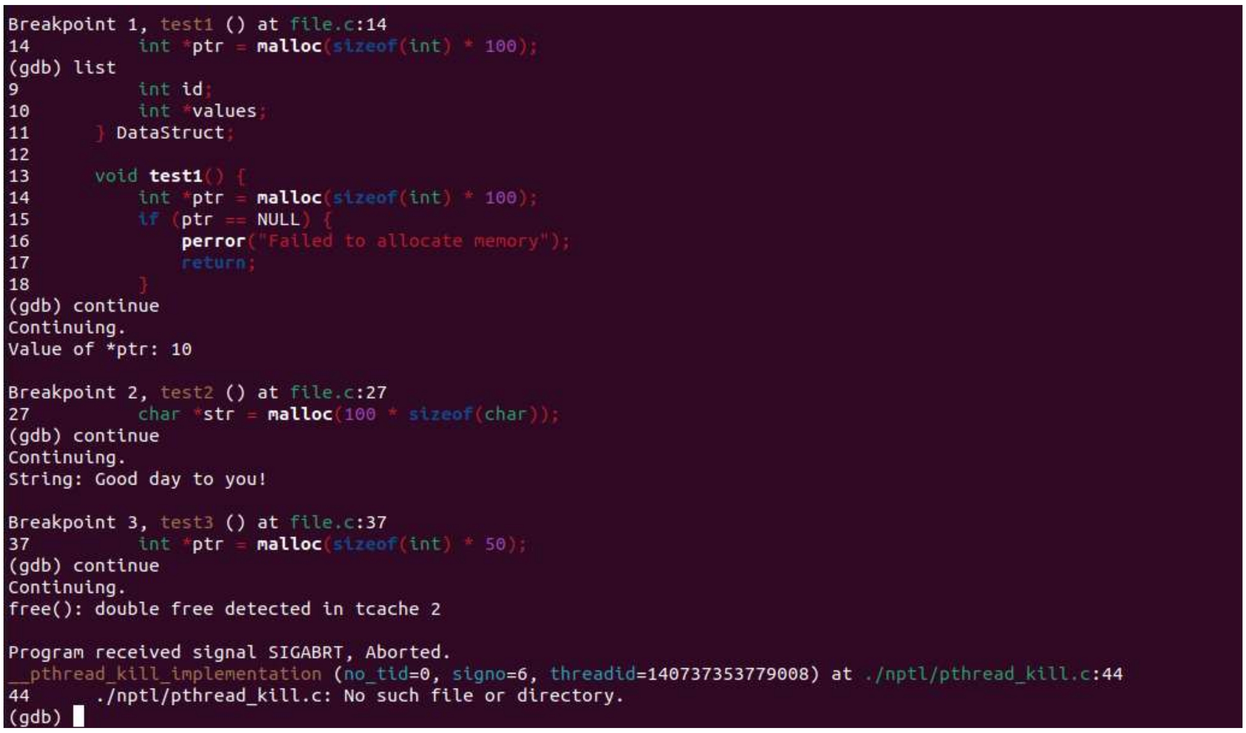
***"list"***



```
somes@somes-ubuntu: ~/Desktop
Continuing.

Program terminated with signal SIGABRT, Aborted.
The program no longer exists.
(gdb) start
Temporary breakpoint 6 at 0x555555555551: file file.c, line 76.
Starting program: /home/somes/Desktop/file
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Temporary breakpoint 6, main () at file.c:76
76          test1();
(gdb) list
71                  }
72              }
73          }
74
75      int main() {
76          test1();
77          test2();
78          test3();
79          test4();
80          test5();
(gdb) continue
Continuing.

Breakpoint 1, test1 () at file.c:14
14          int *ptr = malloc(sizeof(int) * 100);
(gdb) list
9           int id;
10          int *values;
11      } DataStruct;
12
13      void test1() {
14          int *ptr = malloc(sizeof(int) * 100);
15          if (ptr == NULL) {
16              perror("Failed to allocate memory");
17              return;
18          }
(gdb) 
```

Running from one breakpoint to another

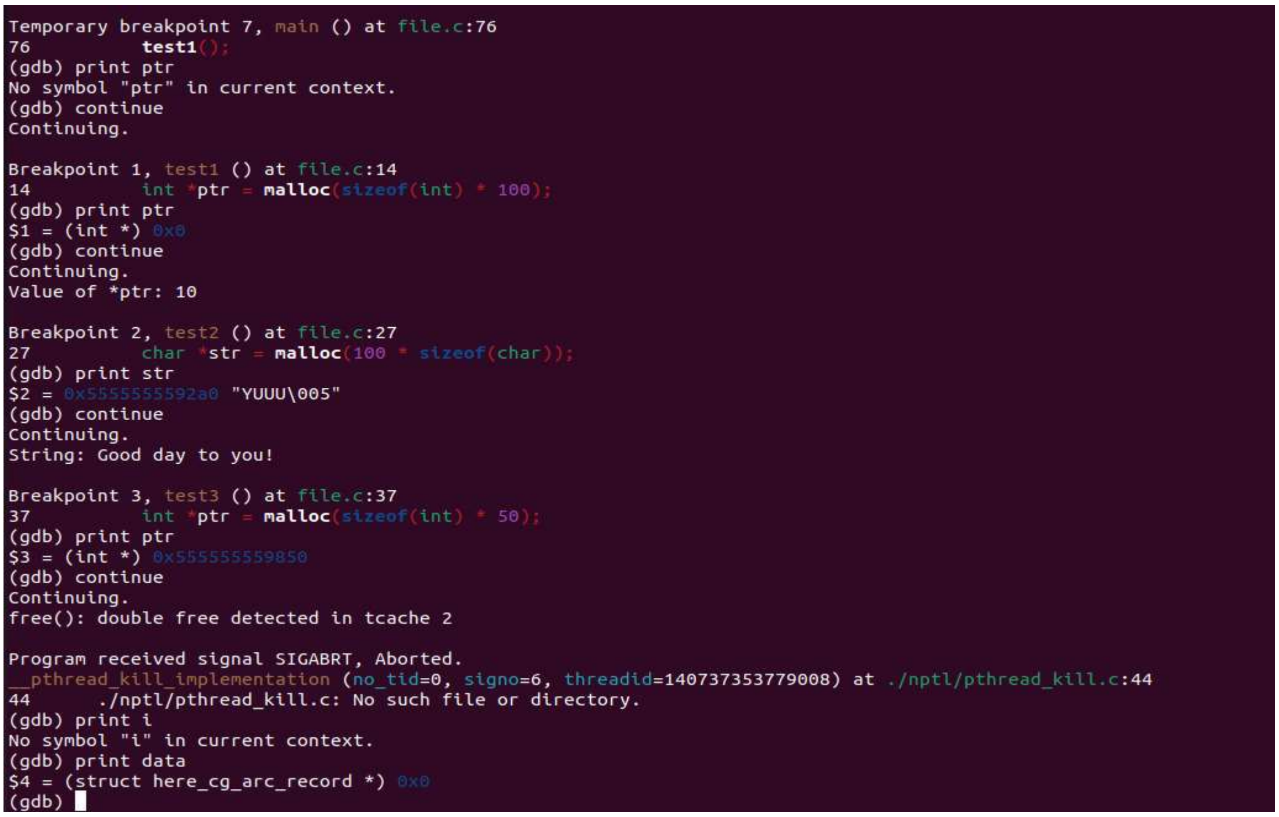
***"continue"***



```
Breakpoint 1, test1 () at file.c:14
14          int *ptr = malloc(sizeof(int) * 100);
(gdb) list
9           int id;
10          int *values;
11      } DataStruct;
12
13      void test1() {
14          int *ptr = malloc(sizeof(int) * 100);
15          if (ptr == NULL) {
16              perror("Failed to allocate memory");
17              return;
18          }
(gdb) continue
Continuing.
Value of *ptr: 10

Breakpoint 2, test2 () at file.c:27
27          char *str = malloc(100 * sizeof(char));
(gdb) continue
Continuing.
String: Good day to you!

Breakpoint 3, test3 () at file.c:37
37          int *ptr = malloc(sizeof(int) * 50);
(gdb) continue
Continuing.
free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (no_tid=0, signo=6, threadid=140737353779008) at ./nptl/pthread_kill.c:44
44      ./nptl/pthread_kill.c: No such file or directory.
(gdb) 
```

Printing the values of variables

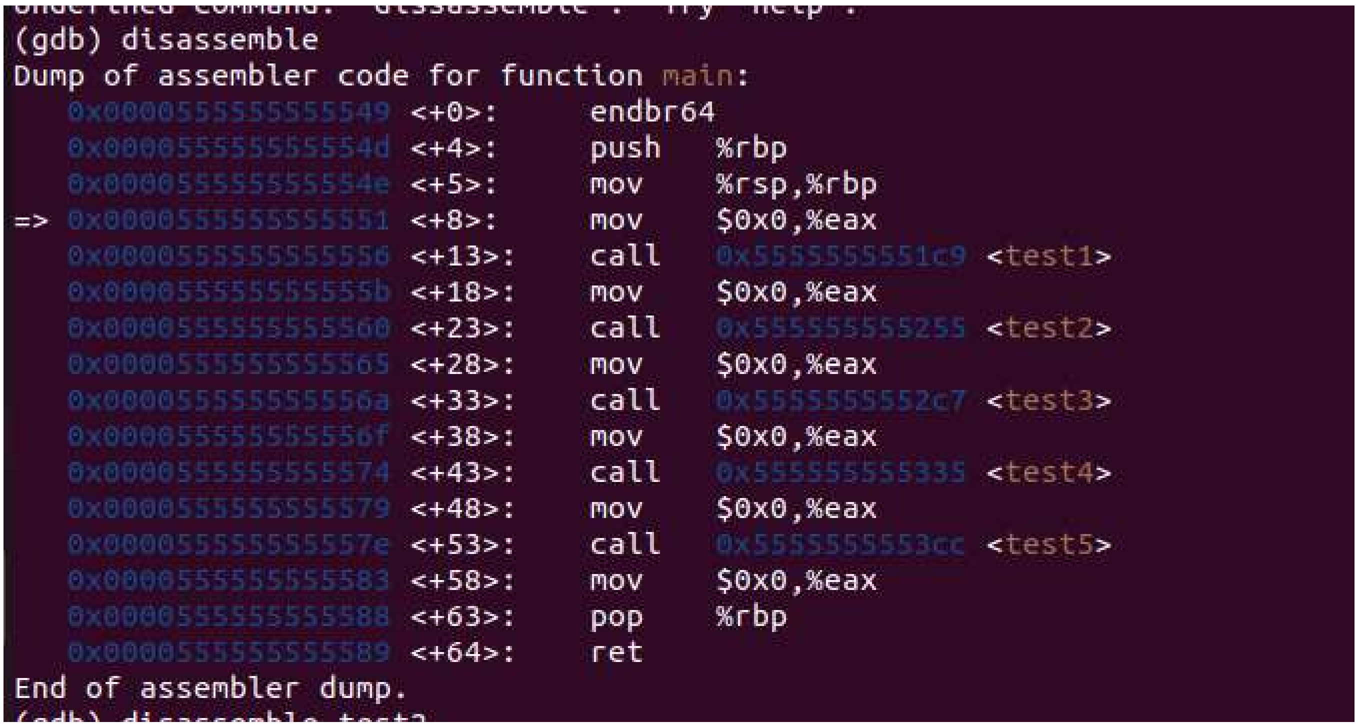
***"print <variable>"***



```
Temporary breakpoint 7, main () at file.c:76
76          test1();
(gdb) print ptr
No symbol "ptr" in current context.
(gdb) continue
Continuing.

Breakpoint 1, test1 () at file.c:14
14          int *ptr = malloc(sizeof(int) * 100);
(gdb) print ptr
$1 = (int *) 0x0
(gdb) continue
Continuing.
Value of *ptr: 10

Breakpoint 2, test2 () at file.c:27
27          char *str = malloc(100 * sizeof(char));
(gdb) print str
$2 = 0x5555555592a0 "YUUU\005"
(gdb) continue
Continuing.
String: Good day to you!

Breakpoint 3, test3 () at file.c:37
37          int *ptr = malloc(sizeof(int) * 50);
(gdb) print ptr
$3 = (int *) 0x555555559850
(gdb) continue
Continuing.
free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (no_tid=0, signo=6, threadid=140737353779008) at ./nptl/pthread_kill.c:44
44      ./nptl/pthread_kill.c: No such file or directory.
(gdb) print i
No symbol "i" in current context.
(gdb) print data
$4 = (struct here_cg_arc_record *) 0x0
(gdb) 
```

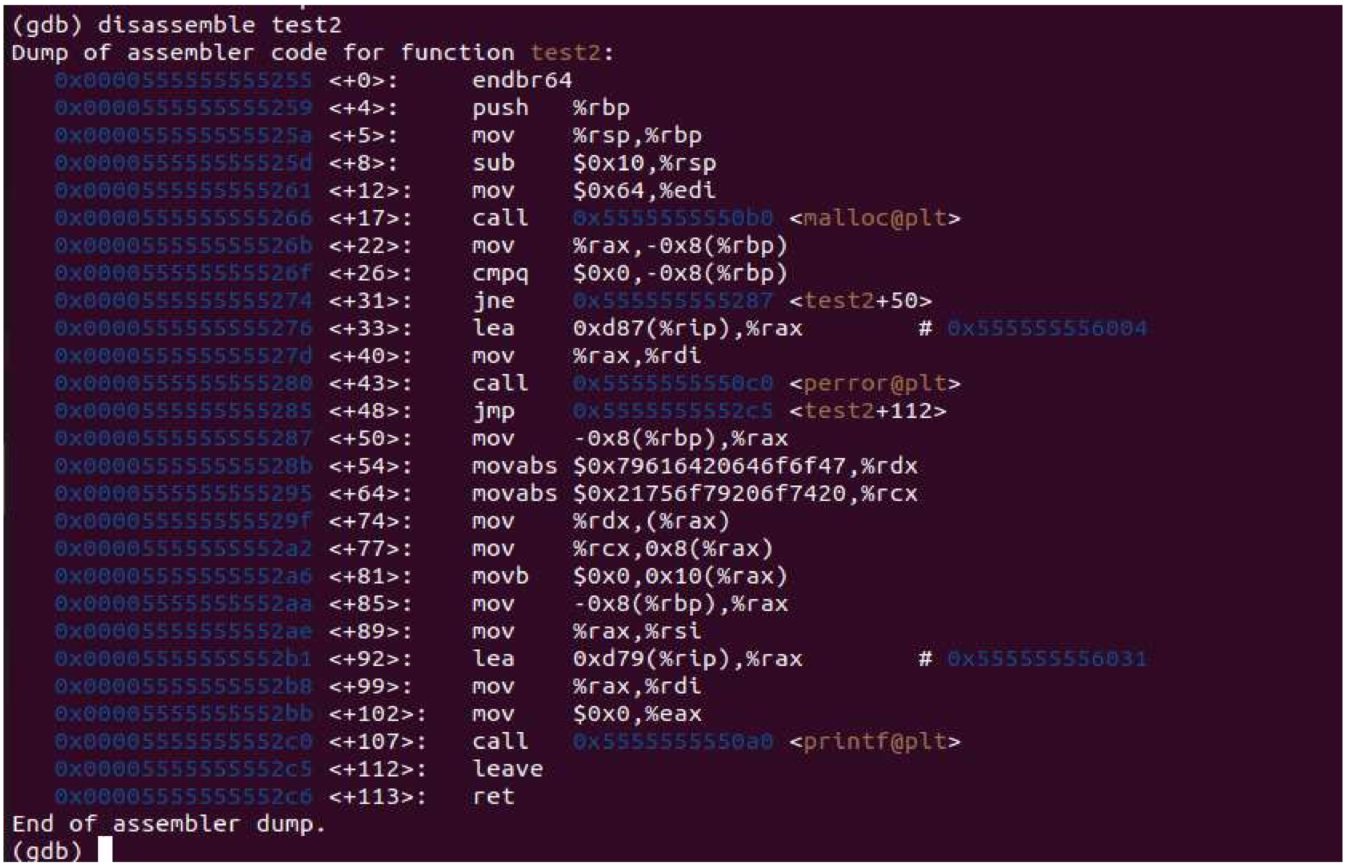
Assembly code

Assembly code of main – “***disassemble***”



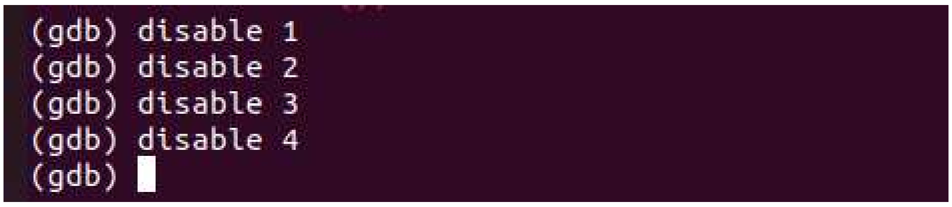
```
(gdb) disassemble
Dump of assembler code for function main:
   0x0000555555555549 <+0>:     endbr64
   0x000055555555554d <+4>:     push   %rbp
   0x000055555555554e <+5>:     mov    %rsp,%rbp
=> 0x0000555555555551 <+8>:     mov    $0x0,%eax
   0x0000555555555556 <+13>:    call   0x5555555551c9 <test1>
   0x000055555555555b <+18>:    mov    $0x0,%eax
   0x0000555555555560 <+23>:    call   0x555555555255 <test2>
   0x0000555555555565 <+28>:    mov    $0x0,%eax
   0x000055555555556a <+33>:    call   0x5555555552c7 <test3>
   0x000055555555556f <+38>:    mov    $0x0,%eax
   0x0000555555555574 <+43>:    call   0x555555555335 <test4>
   0x0000555555555579 <+48>:    mov    $0x0,%eax
   0x000055555555557e <+53>:    call   0x5555555553cc <test5>
   0x0000555555555583 <+58>:    mov    $0x0,%eax
   0x0000555555555588 <+63>:    pop    %rbp
   0x0000555555555589 <+64>:    ret
End of assembler dump.
```

Assembly code for specific function – “***disassemble <function_name>***”



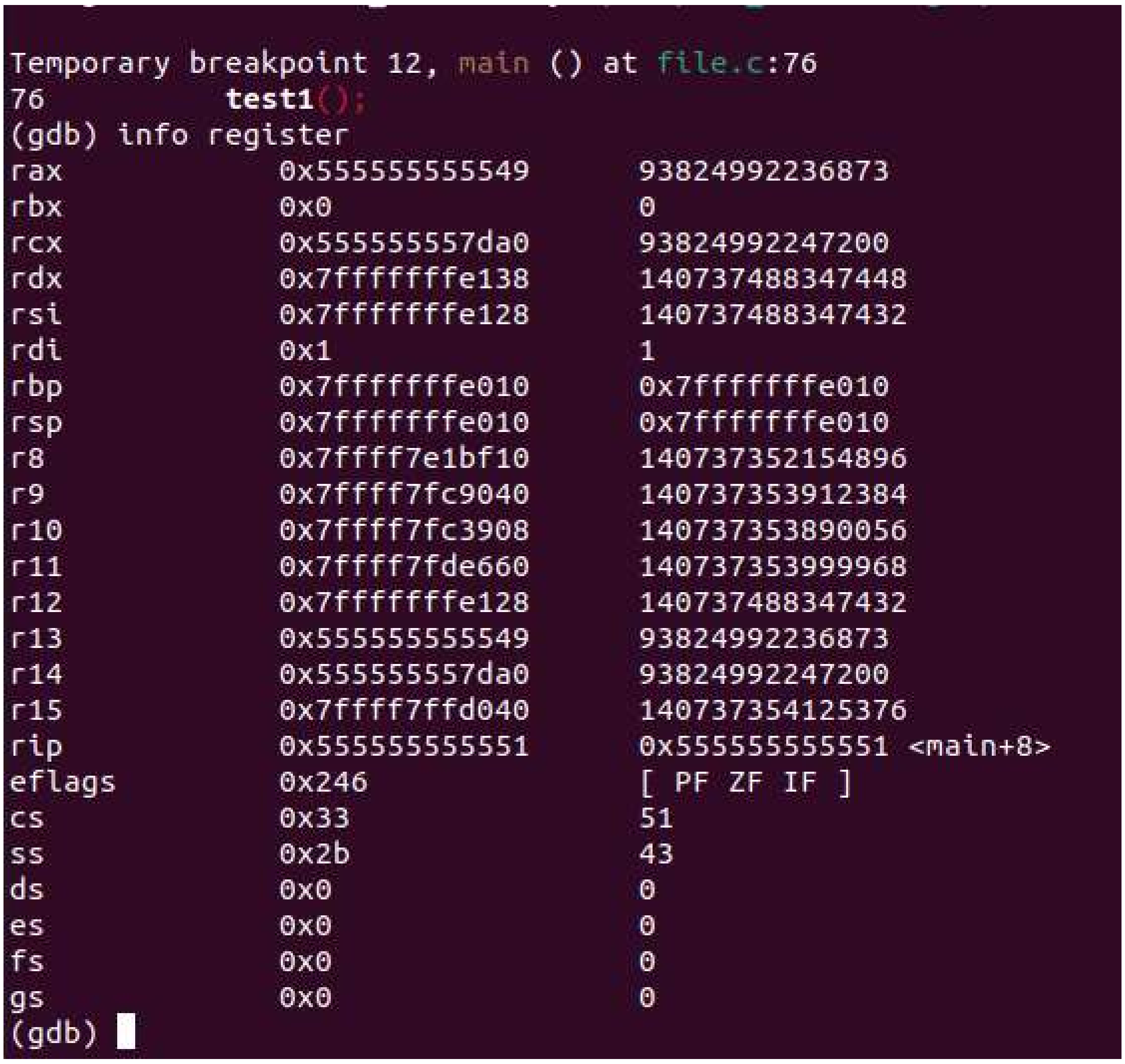
```
(gdb) disassemble test2
Dump of assembler code for function test2:
   0x0000555555555255 <+0>:     endbr64
   0x0000555555555259 <+4>:     push   %rbp
   0x000055555555525a <+5>:     mov    %rsp,%rbp
   0x000055555555525d <+8>:     sub    $0x10,%rsp
   0x0000555555555261 <+12>:    mov    $0x64,%edi
   0x0000555555555266 <+17>:    call   0x5555555550b0 <malloc@plt>
   0x000055555555526b <+22>:    mov    %rax,-0x8(%rbp)
   0x000055555555526f <+26>:    cmpq   $0x0,-0x8(%rbp)
   0x0000555555555274 <+31>:    jne    0x555555555287 <test2+50>
   0x0000555555555276 <+33>:    lea    0xd87(%rip),%rax        # 0x555555556004
   0x000055555555527d <+40>:    mov    %rax,%rdi
   0x0000555555555280 <+43>:    call   0x5555555550c0 <perror@plt>
   0x0000555555555285 <+48>:    jmp    0x5555555552c5 <test2+112>
   0x0000555555555287 <+50>:    mov    -0x8(%rbp),%rax
   0x000055555555528b <+54>:    movabs $0x79616420646f6f47,%rdx
   0x0000555555555295 <+64>:    movabs $0x21756f79206f7420,%rcx
   0x000055555555529f <+74>:    mov    %rdx,(%rax)
   0x00005555555552a2 <+77>:    mov    %rcx,0x8(%rax)
   0x00005555555552a6 <+81>:    movb   $0x0,0x10(%rax)
   0x00005555555552aa <+85>:    mov    -0x8(%rbp),%rax
   0x00005555555552ae <+89>:    mov    %rax,%rsi
   0x00005555555552b1 <+92>:    lea    0xd79(%rip),%rax        # 0x555555556031
   0x00005555555552b8 <+99>:    mov    %rax,%rdi
   0x00005555555552bb <+102>:   mov    $0x0,%eax
   0x00005555555552c0 <+107>:   call   0x5555555550a0 <printf@plt>
   0x00005555555552c5 <+112>:   leave
   0x00005555555552c6 <+113>:   ret
End of assembler dump.
(gdb)
```

Disabling breakpoints – “***disable <breakpoint_number>***”



```
(gdb) disable 1
(gdb) disable 2
(gdb) disable 3
(gdb) disable 4
(gdb) 
```

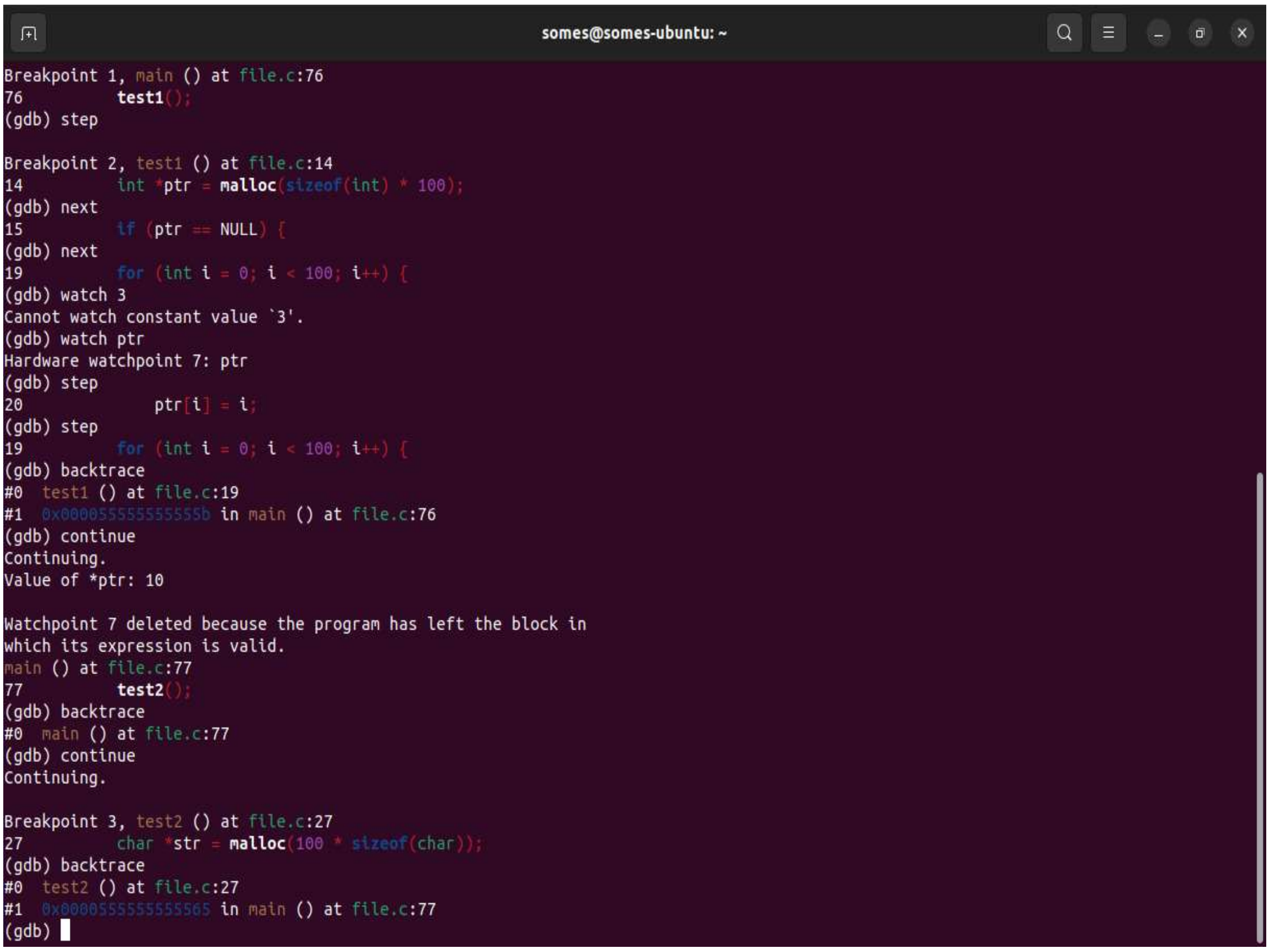
Check register info – “***info register***”



```
Temporary breakpoint 12, main () at file.c:76
76          test1();
(gdb) info register
rax            0x555555555549      93824992236873
rbx            0x0                 0
rcx            0x555555557da0      93824992247200
rdx            0x7fffffffe138      140737488347448
rsi            0x7fffffffe128      140737488347432
rdi            0x1                 1
rbp            0x7fffffffe010      0x7fffffffe010
rsp            0x7fffffffe010      0x7fffffffe010
r8             0x7ffff7e1bf10      140737352154896
r9             0x7ffff7fc9040      140737353912384
r10            0x7ffff7fc3908      140737353890056
r11            0x7ffff7fde660      140737353999968
r12            0x7fffffffe128      140737488347432
r13            0x555555555549      93824992236873
r14            0x555555557da0      93824992247200
r15            0x7ffff7ffd040      140737354125376
rip            0x555555555551      0x555555555551 <main+8>
eflags         0x246               [ PF ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) 
```

Some optional flags

***"step", "next", "backtrace"***



```
Breakpoint 1, main () at file.c:76
76          test1();
(gdb) step

Breakpoint 2, test1 () at file.c:14
14          int *ptr = malloc(sizeof(int) * 100);
(gdb) next
15          if (ptr == NULL) {
(gdb) next
19          for (int i = 0; i < 100; i++) {
(gdb) watch 3
Cannot watch constant value `3'.
(gdb) watch ptr
Hardware watchpoint 7: ptr
(gdb) step
20              ptr[i] = i;
(gdb) step
19          for (int i = 0; i < 100; i++) {
(gdb) backtrace
#0  test1 () at file.c:19
#1  0x000055555555555b in main () at file.c:76
(gdb) continue
Continuing.
Value of *ptr: 10

Watchpoint 7 deleted because the program has left the block in
which its expression is valid.
main () at file.c:77
77          test2();
(gdb) backtrace
#0  main () at file.c:77
(gdb) continue
Continuing.

Breakpoint 3, test2 () at file.c:27
27          char *str = malloc(100 * sizeof(char));
(gdb) backtrace
#0  test2 () at file.c:27
#1  0x0000555555555565 in main () at file.c:77
(gdb) 
```