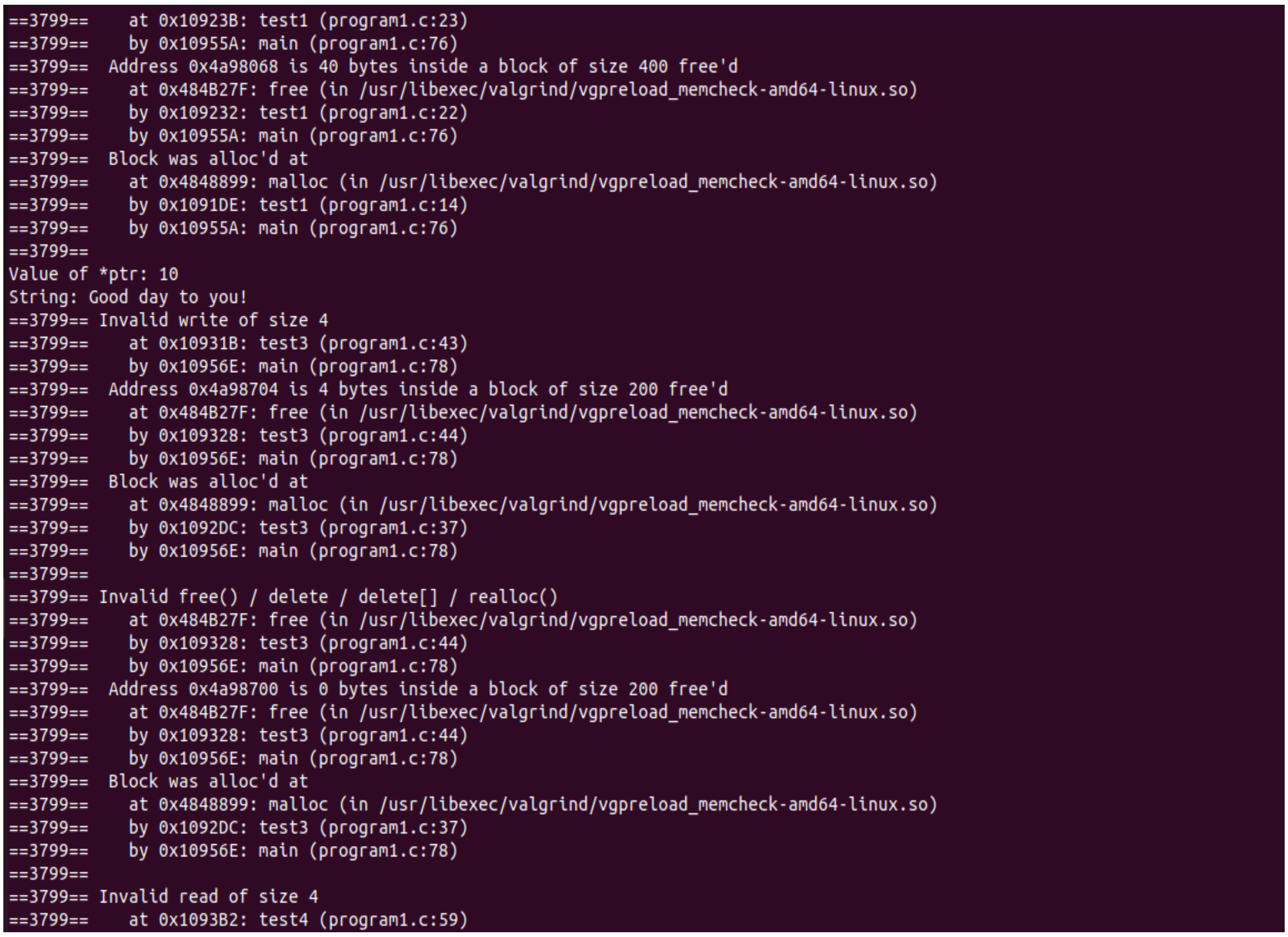
# Module 4 – Assessment
# Code Debugging Tools (GBD and Valgrind)

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

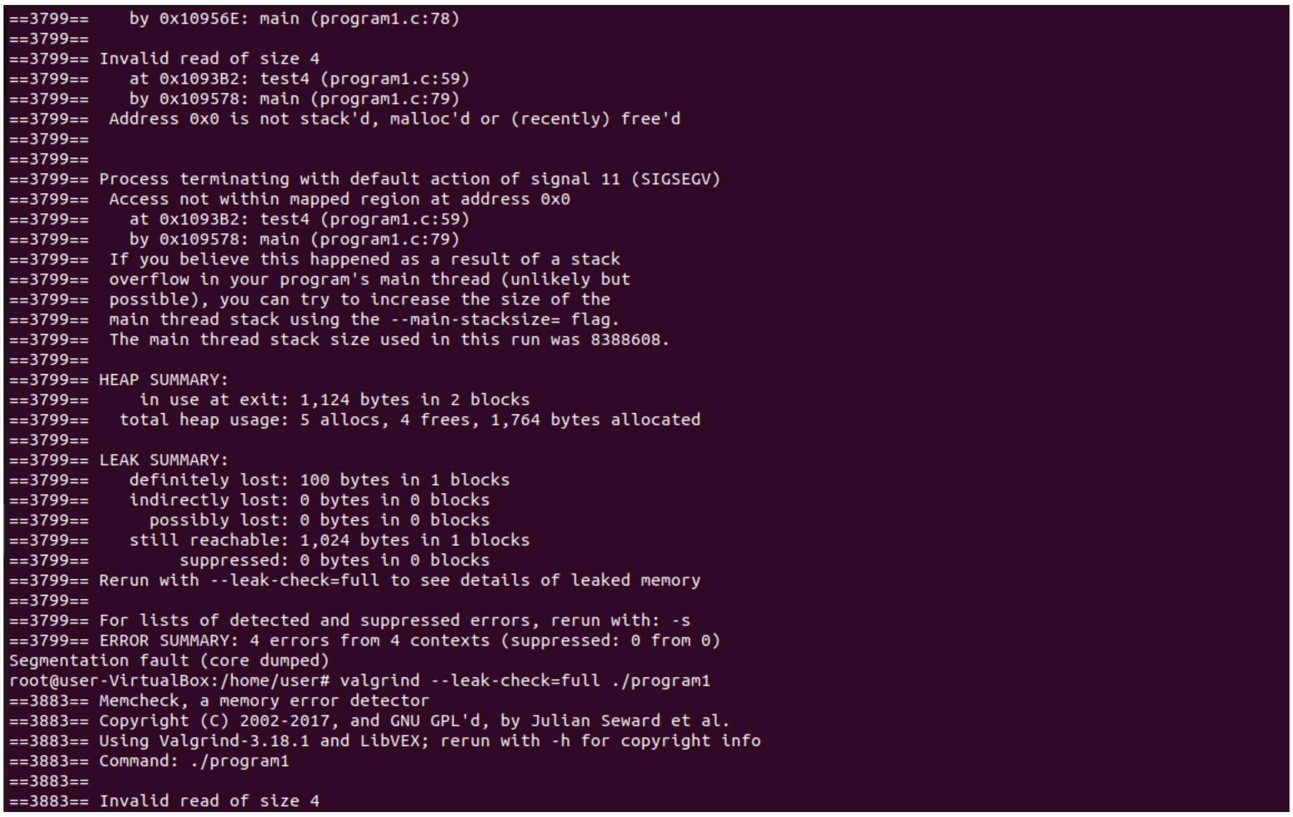
```
gcc -g -o program.out program.c
valgrind --leak-check=full --show-leak-kinds=all ./program.out
```

Optional Valgrind Flags:

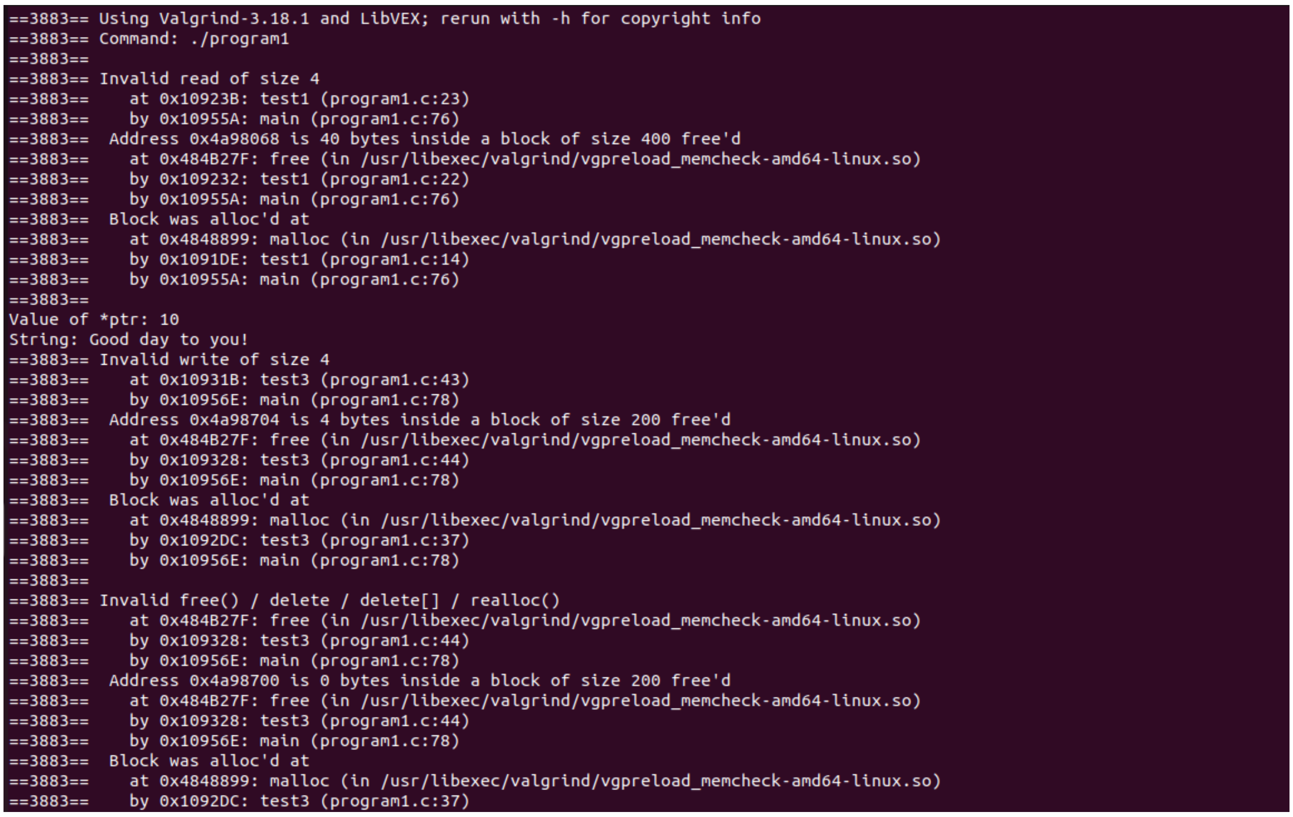
--track-origins=yes: Track the origins of uninitialized values.
--leak-resolution=low|med|high: Control the precision of leak detection.
--log-file=<file>: Redirect output to a specified file.



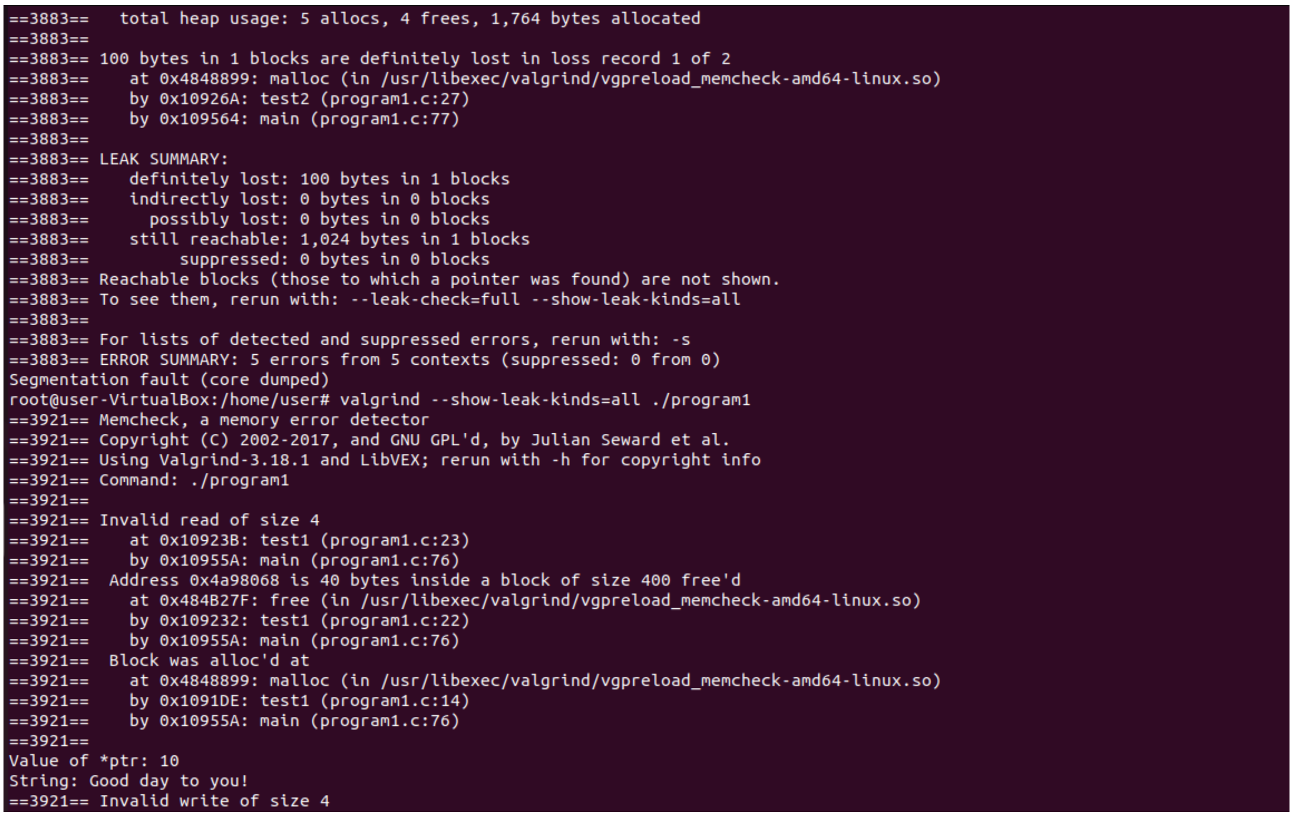
```
==3799==    at 0x10923B: test1 (program1.c:23)
==3799==    by 0x10955A: main (program1.c:76)
==3799==  Address 0x4a98068 is 40 bytes inside a block of size 400 free'd
==3799==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x109232: test1 (program1.c:22)
==3799==    by 0x10955A: main (program1.c:76)
==3799==  Block was alloc'd at
==3799==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x1091DE: test1 (program1.c:14)
==3799==    by 0x10955A: main (program1.c:76)
==3799==
Value of *ptr: 10
String: Good day to you!
==3799== Invalid write of size 4
==3799==    at 0x10931B: test3 (program1.c:43)
==3799==    by 0x10956E: main (program1.c:78)
==3799==  Address 0x4a98704 is 4 bytes inside a block of size 200 free'd
==3799==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x109328: test3 (program1.c:44)
==3799==    by 0x10956E: main (program1.c:78)
==3799==  Block was alloc'd at
==3799==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x1092DC: test3 (program1.c:37)
==3799==    by 0x10956E: main (program1.c:78)
==3799==
==3799== Invalid free() / delete / delete[] / realloc()
==3799==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x109328: test3 (program1.c:44)
==3799==    by 0x10956E: main (program1.c:78)
==3799==  Address 0x4a98700 is 0 bytes inside a block of size 200 free'd
==3799==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x109328: test3 (program1.c:44)
==3799==    by 0x10956E: main (program1.c:78)
==3799==  Block was alloc'd at
==3799==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3799==    by 0x1092DC: test3 (program1.c:37)
==3799==    by 0x10956E: main (program1.c:78)
==3799==
==3799== Invalid read of size 4
==3799==    at 0x1093B2: test4 (program1.c:59)
```



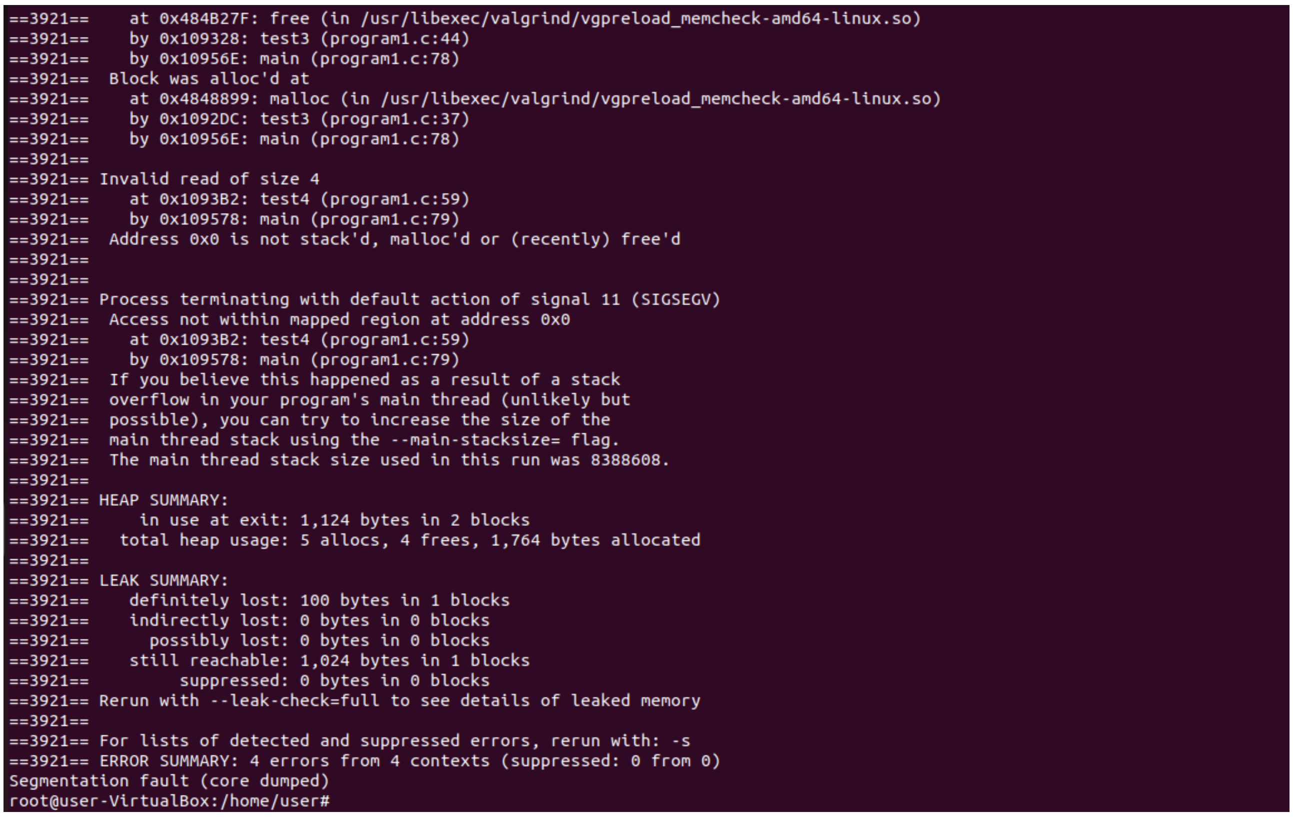
```
==3799==    by 0x10956E: main (program1.c:78)
==3799==
==3799== Invalid read of size 4
==3799==    at 0x1093B2: test4 (program1.c:59)
==3799==    by 0x109578: main (program1.c:79)
==3799==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==3799==
==3799==
==3799== Process terminating with default action of signal 11 (SIGSEGV)
==3799==  Access not within mapped region at address 0x0
==3799==    at 0x1093B2: test4 (program1.c:59)
==3799==    by 0x109578: main (program1.c:79)
==3799==  If you believe this happened as a result of a stack
==3799==  overflow in your program's main thread (unlikely but
==3799==  possible), you can try to increase the size of the
==3799==  main thread stack using the --main-stacksize= flag.
==3799==  The main thread stack size used in this run was 8388608.
==3799==
==3799== HEAP SUMMARY:
==3799==     in use at exit: 1,124 bytes in 2 blocks
==3799==   total heap usage: 5 allocs, 4 frees, 1,764 bytes allocated
==3799==
==3799== LEAK SUMMARY:
==3799==    definitely lost: 100 bytes in 1 blocks
==3799==    indirectly lost: 0 bytes in 0 blocks
==3799==      possibly lost: 0 bytes in 0 blocks
==3799==    still reachable: 1,024 bytes in 1 blocks
==3799==         suppressed: 0 bytes in 0 blocks
==3799== Rerun with --leak-check=full to see details of leaked memory
==3799==
==3799== For lists of detected and suppressed errors, rerun with: -s
==3799== ERROR SUMMARY: 4 errors from 4 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
root@user-VirtualBox:/home/user# valgrind --leak-check=full ./program1
==3883== Memcheck, a memory error detector
==3883== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3883== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==3883== Command: ./program1
==3883==
==3883== Invalid read of size 4
```



```
==3883== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==3883== Command: ./program1
==3883==
==3883== Invalid read of size 4
==3883==    at 0x10923B: test1 (program1.c:23)
==3883==    by 0x10955A: main (program1.c:76)
==3883==  Address 0x4a98068 is 40 bytes inside a block of size 400 free'd
==3883==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x109232: test1 (program1.c:22)
==3883==    by 0x10955A: main (program1.c:76)
==3883==  Block was alloc'd at
==3883==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x1091DE: test1 (program1.c:14)
==3883==    by 0x10955A: main (program1.c:76)
==3883==
Value of *ptr: 10
String: Good day to you!
==3883== Invalid write of size 4
==3883==    at 0x10931B: test3 (program1.c:43)
==3883==    by 0x10956E: main (program1.c:78)
==3883==  Address 0x4a98704 is 4 bytes inside a block of size 200 free'd
==3883==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x109328: test3 (program1.c:44)
==3883==    by 0x10956E: main (program1.c:78)
==3883==  Block was alloc'd at
==3883==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x1092DC: test3 (program1.c:37)
==3883==    by 0x10956E: main (program1.c:78)
==3883==
==3883== Invalid free() / delete / delete[] / realloc()
==3883==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x109328: test3 (program1.c:44)
==3883==    by 0x10956E: main (program1.c:78)
==3883==  Address 0x4a98700 is 0 bytes inside a block of size 200 free'd
==3883==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x109328: test3 (program1.c:44)
==3883==    by 0x10956E: main (program1.c:78)
==3883==  Block was alloc'd at
==3883==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x1092DC: test3 (program1.c:37)
```



```
==3883==   total heap usage: 5 allocs, 4 frees, 1,764 bytes allocated
==3883==
==3883== 100 bytes in 1 blocks are definitely lost in loss record 1 of 2
==3883==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3883==    by 0x10926A: test2 (program1.c:27)
==3883==    by 0x109564: main (program1.c:77)
==3883==
==3883== LEAK SUMMARY:
==3883==    definitely lost: 100 bytes in 1 blocks
==3883==    indirectly lost: 0 bytes in 0 blocks
==3883==      possibly lost: 0 bytes in 0 blocks
==3883==    still reachable: 1,024 bytes in 1 blocks
==3883==         suppressed: 0 bytes in 0 blocks
==3883== Reachable blocks (those to which a pointer was found) are not shown.
==3883== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==3883==
==3883== For lists of detected and suppressed errors, rerun with: -s
==3883== ERROR SUMMARY: 5 errors from 5 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
root@user-VirtualBox:/home/user# valgrind --show-leak-kinds=all ./program1
==3921== Memcheck, a memory error detector
==3921== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3921== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==3921== Command: ./program1
==3921==
==3921== Invalid read of size 4
==3921==    at 0x10923B: test1 (program1.c:23)
==3921==    by 0x10955A: main (program1.c:76)
==3921==  Address 0x4a98068 is 40 bytes inside a block of size 400 free'd
==3921==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3921==    by 0x109232: test1 (program1.c:22)
==3921==    by 0x10955A: main (program1.c:76)
==3921==  Block was alloc'd at
==3921==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3921==    by 0x1091DE: test1 (program1.c:14)
==3921==    by 0x10955A: main (program1.c:76)
==3921==
Value of *ptr: 10
String: Good day to you!
==3921== Invalid write of size 4
```



```
==3921==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3921==    by 0x109328: test3 (program1.c:44)
==3921==    by 0x10956E: main (program1.c:78)
==3921==  Block was alloc'd at
==3921==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3921==    by 0x1092DC: test3 (program1.c:37)
==3921==    by 0x10956E: main (program1.c:78)
==3921==
==3921== Invalid read of size 4
==3921==    at 0x1093B2: test4 (program1.c:59)
==3921==    by 0x109578: main (program1.c:79)
==3921==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==3921==
==3921==
==3921== Process terminating with default action of signal 11 (SIGSEGV)
==3921==  Access not within mapped region at address 0x0
==3921==    at 0x1093B2: test4 (program1.c:59)
==3921==    by 0x109578: main (program1.c:79)
==3921==  If you believe this happened as a result of a stack
==3921==  overflow in your program's main thread (unlikely but
==3921==  possible), you can try to increase the size of the
==3921==  main thread stack using the --main-stacksize= flag.
==3921==  The main thread stack size used in this run was 8388608.
==3921==
==3921== HEAP SUMMARY:
==3921==     in use at exit: 1,124 bytes in 2 blocks
==3921==   total heap usage: 5 allocs, 4 frees, 1,764 bytes allocated
==3921==
==3921== LEAK SUMMARY:
==3921==    definitely lost: 100 bytes in 1 blocks
==3921==    indirectly lost: 0 bytes in 0 blocks
==3921==      possibly lost: 0 bytes in 0 blocks
==3921==    still reachable: 1,024 bytes in 1 blocks
==3921==         suppressed: 0 bytes in 0 blocks
==3921== Rerun with --leak-check=full to see details of leaked memory
==3921==
==3921== For lists of detected and suppressed errors, rerun with: -s
==3921== ERROR SUMMARY: 4 errors from 4 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
root@user-VirtualBox:/home/user#
```

**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.**
**To build the program in Linux - "gcc -g -o <file.out> file.c"**
**Program :**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NUM_STRUCTS 50

typedef struct {
   char *name;
   int id;
   int *values;
} DataStruct;

void test1() {
   int *ptr = malloc(sizeof(int) * 100);
   if (ptr == NULL) {
      perror("Failed to allocate memory");
      return;
   }
   for (int i = 0; i < 100; i++) {
      ptr[i] = i;
   }
   free(ptr);
   printf("Value of *ptr: %d\n", ptr[10]);
}

void test2() {
   char *str = malloc(100 * sizeof(char));
   if (str == NULL) {
      perror("Failed to allocate memory");
      return;
   }
   strcpy(str, "Good day to you!");
   printf("String: %s\n", str);
}

void test3() {
   int *ptr = malloc(sizeof(int) * 50);
   if (ptr == NULL) {
      perror("Failed to allocate memory");
      return;
   }
   for (int i = 0; i < 2; i++) {
      ptr[i] = i * 2;
      free(ptr);
   }
}
```

```c
void test4() {
    int *ptr = malloc(sizeof(int) * 10);
    if (ptr == NULL) {
        perror("Failed to allocate memory");
        return;
    }
    for (int i = 0; i < 10; i++) {
        ptr[i] = i * 3;
    }
    free(ptr);
    ptr = NULL;
    printf("Value of *ptr: %d\n", *ptr);
}

void test5() {
    DataStruct *data = malloc(NUM_STRUCTS * sizeof(DataStruct));
    for (int i = 0; i < NUM_STRUCTS; i++) {
        data[i].name = malloc(50 * sizeof(char));
        strcpy(data[i].name, "Example Name");
        data[i].id = rand() % 1000;
        data[i].values = malloc(10 * sizeof(int));
        for (int j = 0; j < 10; j++) {
            data[i].values[j] = rand() % 100;
        }
    }
}

int main() {
    test1();
    test2();
    test3();
    test4();
    test5();

    return 0;
}
```

```
gdb ./program.out
```

Set Breakpoints:

```
gdb

(gdb) break test1
(gdb) break test2
(gdb) break test3
(gdb) break test4
(gdb) break test5
```

Run the Program:

```
 (gdb) run
```

List the Code Around the Breakpoint:

```
(gdb) list
```

Step Through the Program:
 (gdb) next
(gdb) continue

Print the Value of Variables:
(gdb) print ptr
(gdb) print str

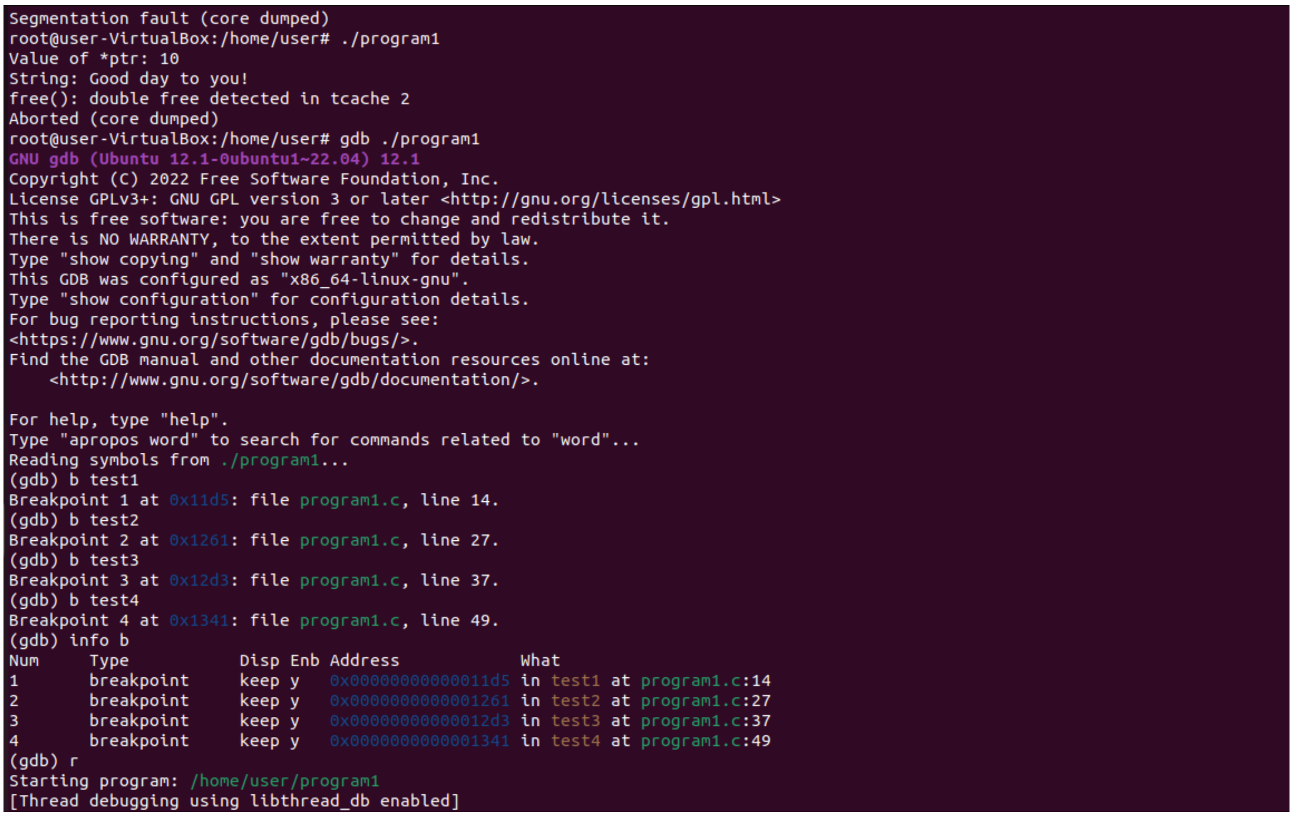
Check Assembly Code:
 (gdb) disassemble

Disable Breakpoints:
 (gdb) disable <breakpoint number>

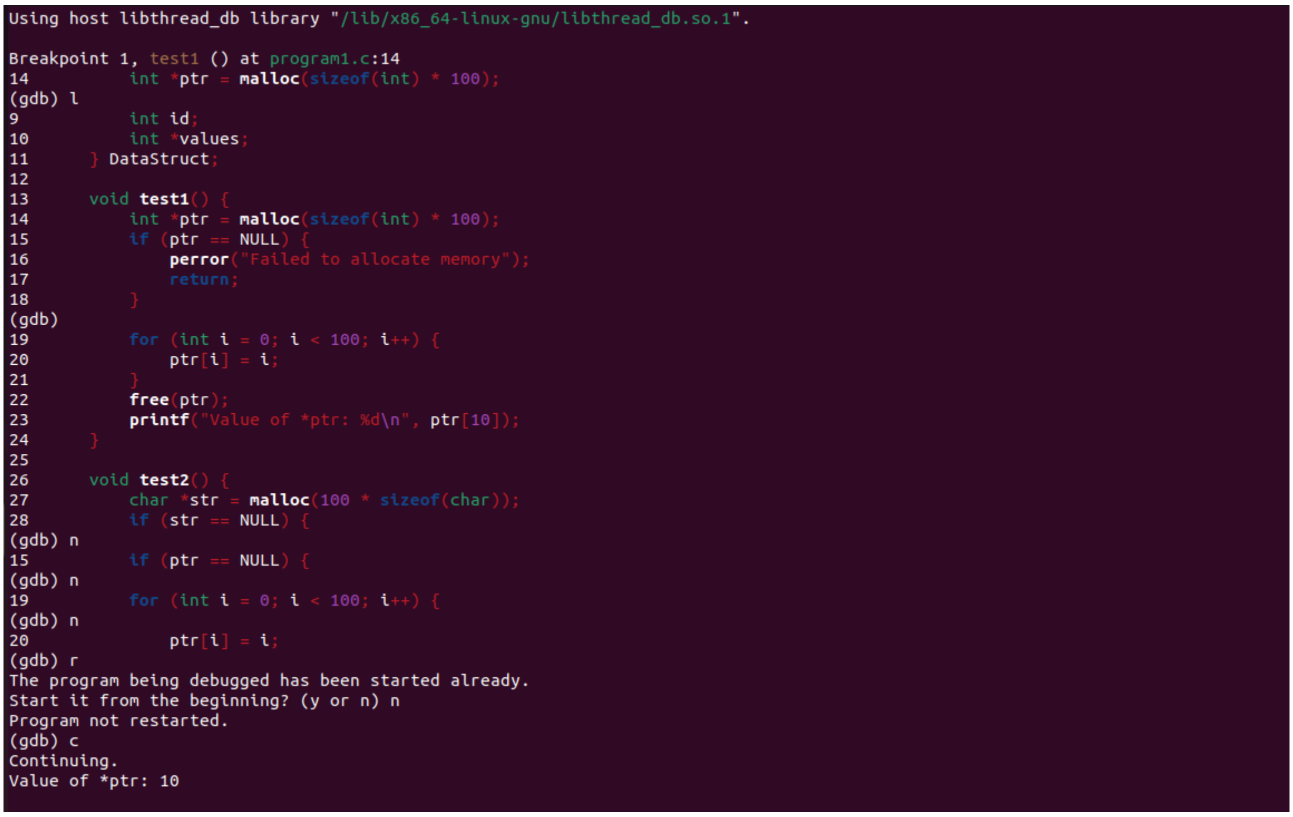
Check Registers Info:
(gdb) info registers

Optional GDB Flags:

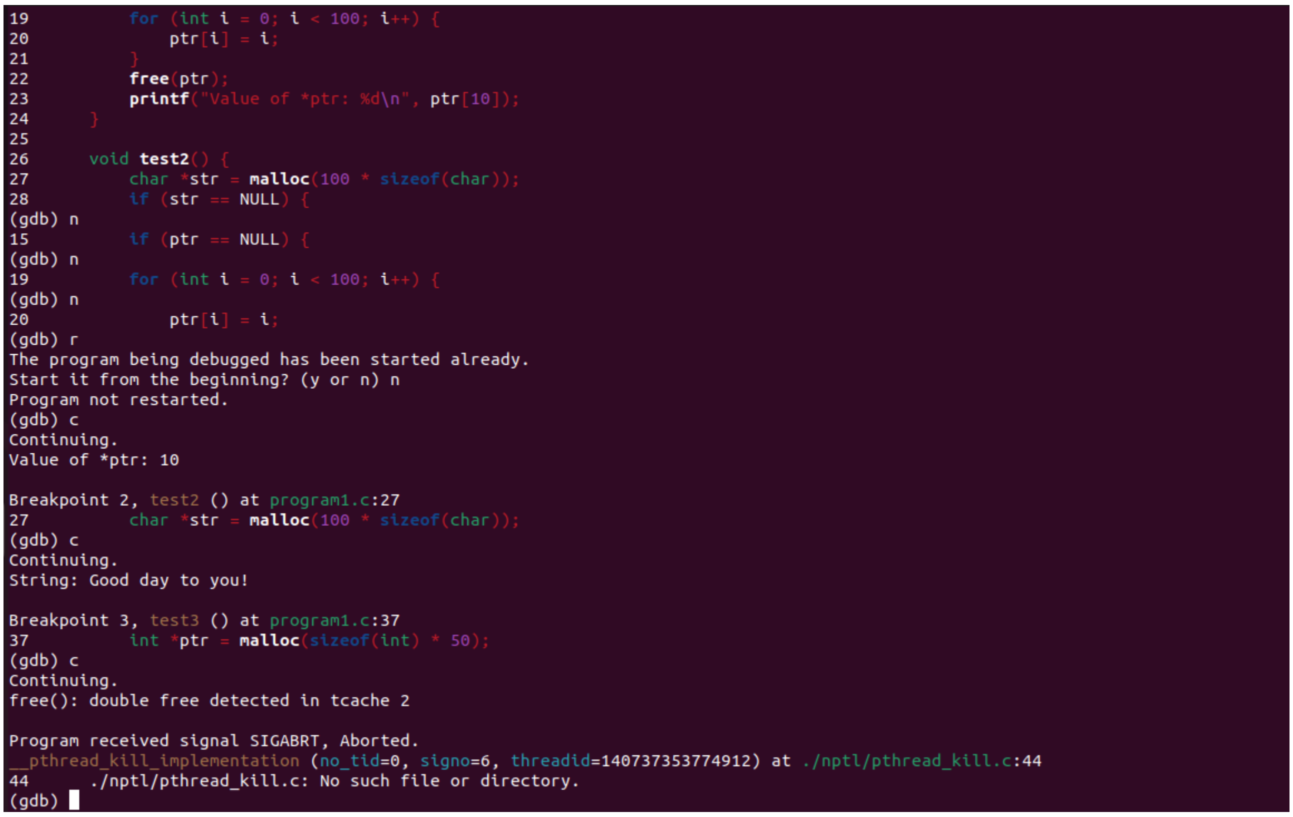
-q: Start GDB in quiet mode.
-ex <command>: Run GDB command on startup.
-tui: Start GDB with Text User Interface.



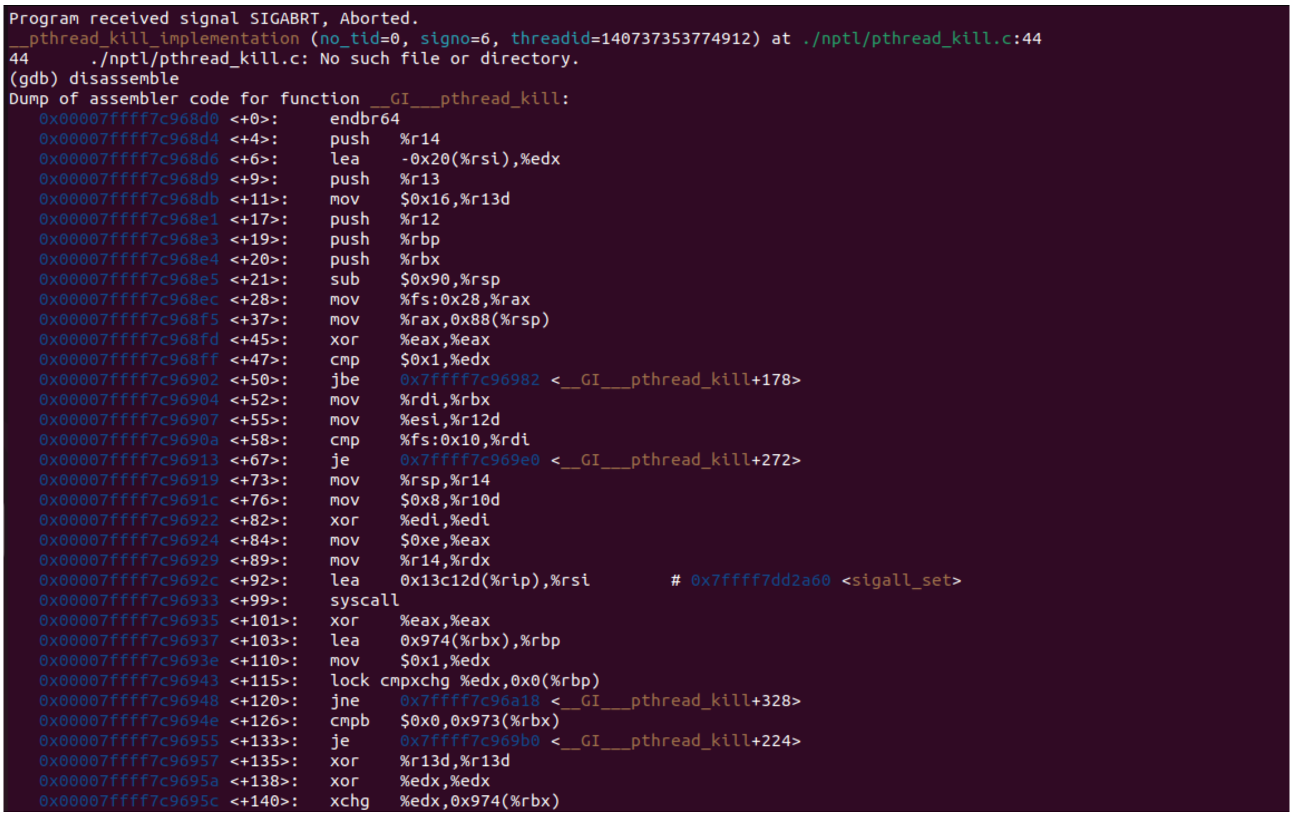
```
Segmentation fault (core dumped)
root@user-VirtualBox:/home/user# ./program1
Value of *ptr: 10
String: Good day to you!
free(): double free detected in tcache 2
Aborted (core dumped)
root@user-VirtualBox:/home/user# gdb ./program1
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free 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 ./program1...
(gdb) b test1
Breakpoint 1 at 0x11d5: file program1.c, line 14.
(gdb) b test2
Breakpoint 2 at 0x1261: file program1.c, line 27.
(gdb) b test3
Breakpoint 3 at 0x12d3: file program1.c, line 37.
(gdb) b test4
Breakpoint 4 at 0x1341: file program1.c, line 49.
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011d5 in test1 at program1.c:14
2       breakpoint     keep y   0x0000000000001261 in test2 at program1.c:27
3       breakpoint     keep y   0x00000000000012d3 in test3 at program1.c:37
4       breakpoint     keep y   0x0000000000001341 in test4 at program1.c:49
(gdb) r
Starting program: /home/user/program1
[Thread debugging using libthread_db enabled]
```



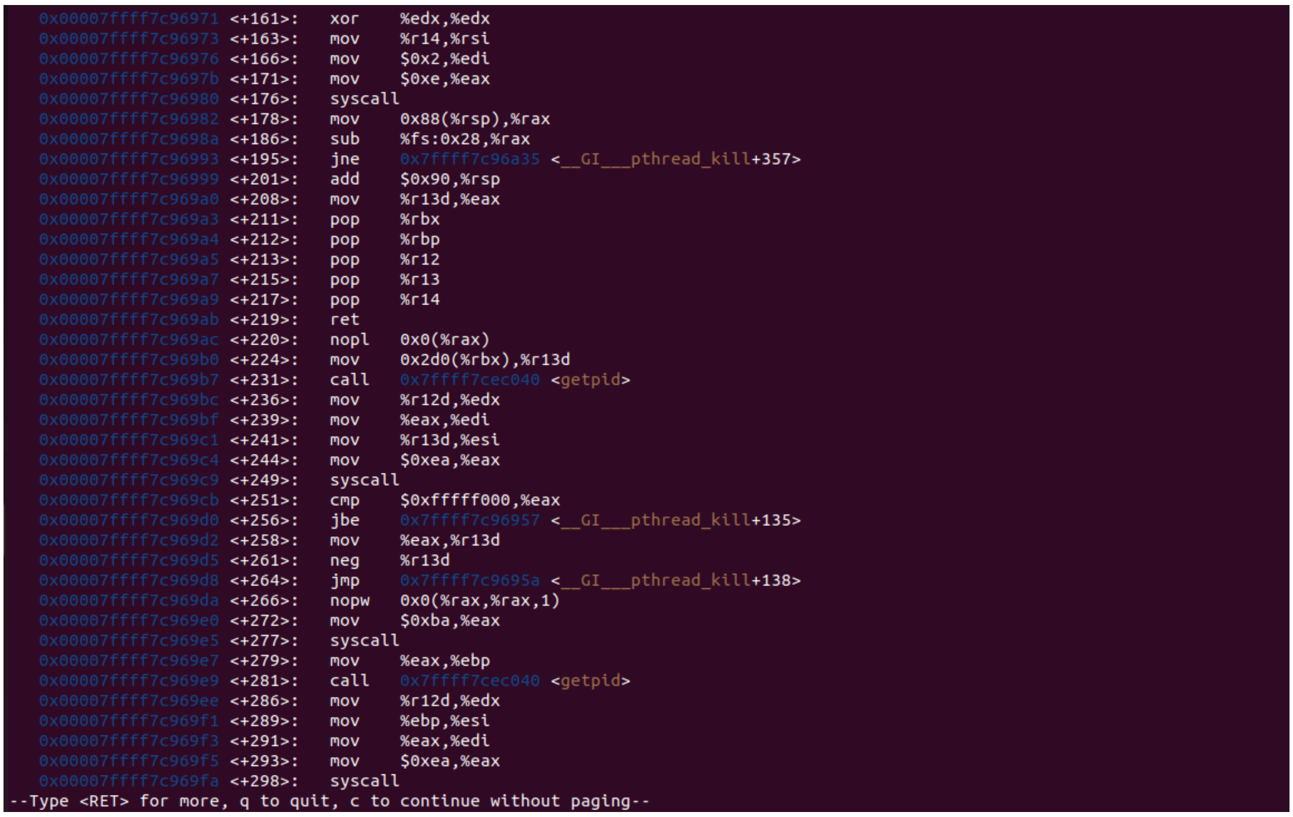
```
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, test1 () at program1.c:14
14          int *ptr = malloc(sizeof(int) * 100);
(gdb) l
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)
19          for (int i = 0; i < 100; i++) {
20              ptr[i] = i;
21          }
22          free(ptr);
23          printf("Value of *ptr: %d\n", ptr[10]);
24      }
25
26      void test2() {
27          char *str = malloc(100 * sizeof(char));
28          if (str == NULL) {
(gdb) n
15          if (ptr == NULL) {
(gdb) n
19          for (int i = 0; i < 100; i++) {
(gdb) n
20              ptr[i] = i;
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) c
Continuing.
Value of *ptr: 10
```



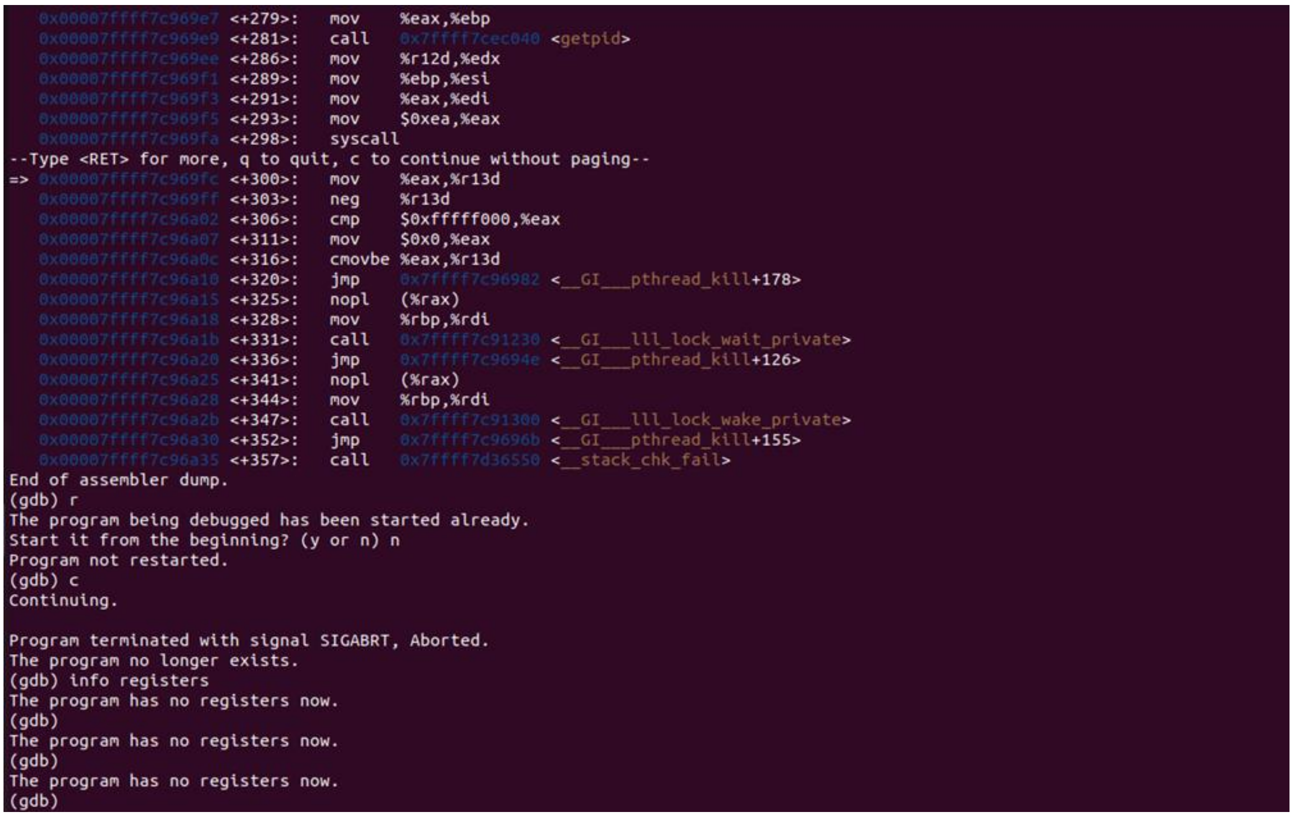
```
19          for (int i = 0; i < 100; i++) {
20              ptr[i] = i;
21          }
22          free(ptr);
23          printf("Value of *ptr: %d\n", ptr[10]);
24      }
25
26      void test2() {
27          char *str = malloc(100 * sizeof(char));
28          if (str == NULL) {
(gdb) n
15          if (ptr == NULL) {
(gdb) n
19          for (int i = 0; i < 100; i++) {
(gdb) n
20              ptr[i] = i;
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) c
Continuing.
Value of *ptr: 10

Breakpoint 2, test2 () at program1.c:27
27          char *str = malloc(100 * sizeof(char));
(gdb) c
Continuing.
String: Good day to you!

Breakpoint 3, test3 () at program1.c:37
37          int *ptr = malloc(sizeof(int) * 50);
(gdb) c
Continuing.
free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (no_tid=0, signo=6, threadid=140737353774912) at ./nptl/pthread_kill.c:44
44      ./nptl/pthread_kill.c: No such file or directory.
(gdb) 
```



```
Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (no_tid=0, signo=6, threadid=140737353774912) at ./nptl/pthread_kill.c:44
44      ./nptl/pthread_kill.c: No such file or directory.
(gdb) disassemble
Dump of assembler code for function __GI___pthread_kill:
   0x00007ffff7c968d0 <+0>:     endbr64
   0x00007ffff7c968d4 <+4>:     push   %r14
   0x00007ffff7c968d6 <+6>:     lea    -0x20(%rsi),%edx
   0x00007ffff7c968d9 <+9>:     push   %r13
   0x00007ffff7c968db <+11>:    mov    $0x16,%r13d
   0x00007ffff7c968e1 <+17>:    push   %r12
   0x00007ffff7c968e3 <+19>:    push   %rbp
   0x00007ffff7c968e4 <+20>:    push   %rbx
   0x00007ffff7c968e5 <+21>:    sub    $0x90,%rsp
   0x00007ffff7c968ec <+28>:    mov    %fs:0x28,%rax
   0x00007ffff7c968f5 <+37>:    mov    %rax,0x88(%rsp)
   0x00007ffff7c968fd <+45>:    xor    %eax,%eax
   0x00007ffff7c968ff <+47>:    cmp    $0x1,%edx
   0x00007ffff7c96902 <+50>:    jbe    0x7ffff7c96982 <__GI___pthread_kill+178>
   0x00007ffff7c96904 <+52>:    mov    %rdi,%rbx
   0x00007ffff7c96907 <+55>:    mov    %esi,%r12d
   0x00007ffff7c9690a <+58>:    cmp    %fs:0x10,%rdi
   0x00007ffff7c96913 <+67>:    je     0x7ffff7c969e0 <__GI___pthread_kill+272>
   0x00007ffff7c96919 <+73>:    mov    %rsp,%r14
   0x00007ffff7c9691c <+76>:    mov    $0x8,%r10d
   0x00007ffff7c96922 <+82>:    xor    %edi,%edi
   0x00007ffff7c96924 <+84>:    mov    $0xe,%eax
   0x00007ffff7c96929 <+89>:    mov    %r14,%rdx
   0x00007ffff7c9692c <+92>:    lea    0x13c12d(%rip),%rsi        # 0x7ffff7dd2a60 <sigall_set>
   0x00007ffff7c96933 <+99>:    syscall
   0x00007ffff7c96935 <+101>:   xor    %eax,%eax
   0x00007ffff7c96937 <+103>:   lea    0x974(%rbx),%rbp
   0x00007ffff7c9693e <+110>:   mov    $0x1,%edx
   0x00007ffff7c96943 <+115>:   lock cmpxchg %edx,0x0(%rbp)
   0x00007ffff7c96948 <+120>:   jne    0x7ffff7c96a18 <__GI___pthread_kill+328>
   0x00007ffff7c9694e <+126>:   cmpb   $0x0,0x973(%rbx)
   0x00007ffff7c96955 <+133>:   je     0x7ffff7c969b0 <__GI___pthread_kill+224>
   0x00007ffff7c96957 <+135>:   xor    %r13d,%r13d
   0x00007ffff7c9695a <+138>:   xor    %edx,%edx
   0x00007ffff7c9695c <+140>:   xchg   %edx,0x974(%rbx)
```



```
   0x00007ffff7c96971 <+161>:   xor    %edx,%edx
   0x00007ffff7c96973 <+163>:   mov    %r14,%rsi
   0x00007ffff7c96976 <+166>:   mov    $0x2,%edi
   0x00007ffff7c9697b <+171>:   mov    $0xe,%eax
   0x00007ffff7c96980 <+176>:   syscall
   0x00007ffff7c96982 <+178>:   mov    0x88(%rsp),%rax
   0x00007ffff7c9698a <+186>:   sub    %fs:0x28,%rax
   0x00007ffff7c96993 <+195>:   jne    0x7ffff7c96a35 <__GI___pthread_kill+357>
   0x00007ffff7c96999 <+201>:   add    $0x90,%rsp
   0x00007ffff7c969a0 <+208>:   mov    %r13d,%eax
   0x00007ffff7c969a3 <+211>:   pop    %rbx
   0x00007ffff7c969a4 <+212>:   pop    %rbp
   0x00007ffff7c969a5 <+213>:   pop    %r12
   0x00007ffff7c969a7 <+215>:   pop    %r13
   0x00007ffff7c969a9 <+217>:   pop    %r14
   0x00007ffff7c969ab <+219>:   ret
   0x00007ffff7c969ac <+220>:   nopl   0x0(%rax)
   0x00007ffff7c969b0 <+224>:   mov    0x2d0(%rbx),%r13d
   0x00007ffff7c969b7 <+231>:   call   0x7ffff7cec040 <getpid>
   0x00007ffff7c969bc <+236>:   mov    %r12d,%edx
   0x00007ffff7c969bf <+239>:   mov    %eax,%edi
   0x00007ffff7c969c1 <+241>:   mov    %r13d,%esi
   0x00007ffff7c969c4 <+244>:   mov    $0xea,%eax
   0x00007ffff7c969c9 <+249>:   syscall
   0x00007ffff7c969cb <+251>:   cmp    $0xfffff000,%eax
   0x00007ffff7c969d0 <+256>:   jbe    0x7ffff7c96957 <__GI___pthread_kill+135>
   0x00007ffff7c969d2 <+258>:   mov    %eax,%r13d
   0x00007ffff7c969d5 <+261>:   neg    %r13d
   0x00007ffff7c969d8 <+264>:   jmp    0x7ffff7c9695a <__GI___pthread_kill+138>
   0x00007ffff7c969da <+266>:   nopw   0x0(%rax,%rax,1)
   0x00007ffff7c969e0 <+272>:   mov    $0xba,%eax
   0x00007ffff7c969e5 <+277>:   syscall
   0x00007ffff7c969e7 <+279>:   mov    %eax,%ebp
   0x00007ffff7c969e9 <+281>:   call   0x7ffff7cec040 <getpid>
   0x00007ffff7c969ee <+286>:   mov    %r12d,%edx
   0x00007ffff7c969f1 <+289>:   mov    %ebp,%esi
   0x00007ffff7c969f3 <+291>:   mov    %eax,%edi
   0x00007ffff7c969f5 <+293>:   mov    $0xea,%eax
   0x00007ffff7c969fa <+298>:   syscall
--Type <RET> for more, q to quit, c to continue without paging--
```



```
   0x00007ffff7c969e7 <+279>:   mov    %eax,%ebp
   0x00007ffff7c969e9 <+281>:   call   0x7ffff7cec040 <getpid>
   0x00007ffff7c969ee <+286>:   mov    %r12d,%edx
   0x00007ffff7c969f1 <+289>:   mov    %ebp,%esi
   0x00007ffff7c969f3 <+291>:   mov    %eax,%edi
   0x00007ffff7c969f5 <+293>:   mov    $0xea,%eax
   0x00007ffff7c969fa <+298>:   syscall
--Type <RET> for more, q to quit, c to continue without paging--
=> 0x00007ffff7c969fc <+300>:   mov    %eax,%r13d
   0x00007ffff7c969ff <+303>:   neg    %r13d
   0x00007ffff7c96a02 <+306>:   cmp    $0xfffff000,%eax
   0x00007ffff7c96a07 <+311>:   mov    $0x0,%eax
   0x00007ffff7c96a0c <+316>:   cmovbe %eax,%r13d
   0x00007ffff7c96a10 <+320>:   jmp    0x7ffff7c96982 <__GI___pthread_kill+178>
   0x00007ffff7c96a15 <+325>:   nopl   (%rax)
   0x00007ffff7c96a18 <+328>:   mov    %rbp,%rdi
   0x00007ffff7c96a1b <+331>:   call   0x7ffff7c91230 <__GI___lll_lock_wait_private>
   0x00007ffff7c96a20 <+336>:   jmp    0x7ffff7c9694e <__GI___pthread_kill+126>
   0x00007ffff7c96a25 <+341>:   nopl   (%rax)
   0x00007ffff7c96a28 <+344>:   mov    %rbp,%rdi
   0x00007ffff7c96a2b <+347>:   call   0x7ffff7c91300 <__GI___lll_lock_wake_private>
   0x00007ffff7c96a30 <+352>:   jmp    0x7ffff7c9696b <__GI___pthread_kill+155>
   0x00007ffff7c96a35 <+357>:   call   0x7ffff7d36550 <__stack_chk_fail>
End of assembler dump.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Program not restarted.
(gdb) c
Continuing.

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