Module 4 – Assessment

Code Debugging Tools (GBD and Valgrind)

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

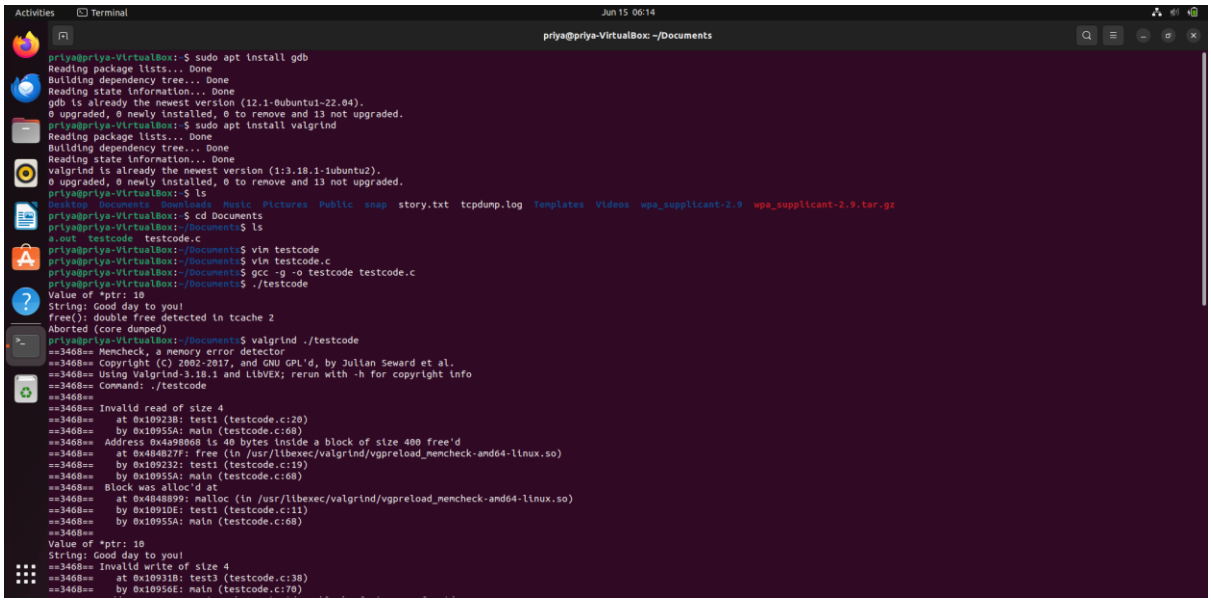
First of all I've created a file testcode.c in Vim editor, copy pasted the given code and saved it. And then I executed the code using valgrind by giving the following commands:

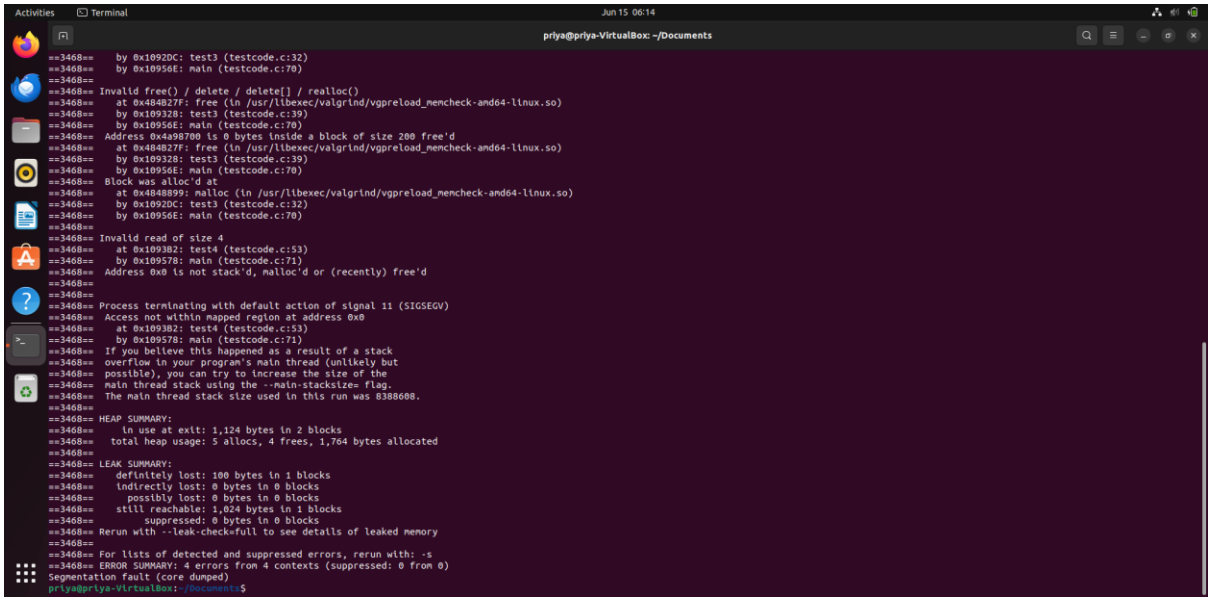
vim testcode.c

gcc –g –o testcode testcode.c

./testcode

valgrind ./testcode





Optional flags that I used are:

valgrind --leak-check=full ./testcode

```
priya@priya-VirtualBox:~/Documents$ valgrind --leak-check=full  ./testcode
==3546== Memcheck, a memory error detector
==3546== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3546== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==3546== Command: ./testcode
==3546==
==3546== Invalid read of size 4
==3546==    at 0x10923B: test1 (testcode.c:20)
==3546==    by 0x10955A: main (testcode.c:68)
==3546==  Address 0x4a98068 is 40 bytes inside a block of size 400 free'd
==3546==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3546==    by 0x109232: test1 (testcode.c:19)
==3546==    by 0x10955A: main (testcode.c:68)
==3546==  Block was alloc'd at
==3546==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
Ubuntu Software
==3546==    by 0x1091DE: test1 (testcode.c:11)
==3546==    by 0x10955A: main (testcode.c:68)
==3546==
Value of *ptr: 10
String: Good day to you!
==3546== Invalid write of size 4
==3546==    at 0x10931B: test3 (testcode.c:38)
==3546==    by 0x10956E: main (testcode.c:70)
==3546==  Address 0x4a98704 is 4 bytes inside a block of size 200 free'd
==3546==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3546==    by 0x109328: test3 (testcode.c:39)
==3546==    by 0x10956E: main (testcode.c:70)
==3546==  Block was alloc'd at
==3546==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3546==    by 0x1092DC: test3 (testcode.c:32)
==3546==    by 0x10956E: main (testcode.c:70)
==3546==
==3546== Invalid free() / delete / delete[] / realloc()
==3546==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3546==    by 0x109328: test3 (testcode.c:39)
==3546==    by 0x10956E: main (testcode.c:70)
==3546==  Address 0x4a98700 is 0 bytes inside a block of size 200 free'd
==3546==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3546==    by 0x109328: test3 (testcode.c:39)
==3546==    by 0x10956E: main (testcode.c:70)
==3546==  Block was alloc'd at
==3546==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==3546==    by 0x1092DC: test3 (testcode.c:32)
==3546==    by 0x10956E: main (testcode.c:70)
```

valgrind --show-leakkinds=all ./testcode

```
priya@priya-VirtualBox:~/Documents$ valgrind --show-leak-kinds=all ./testcode
==4510== Memcheck, a memory error detector
==4510== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4510== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==4510== Command: ./testcode
==4510==
==4510== Invalid read of size 4
==4510==    at 0x10923B: test1 (testcode.c:20)
==4510==    by 0x10955A: main (testcode.c:68)
==4510==  Address 0x4a98068 is 40 bytes inside a block of size 400 free'd
==4510==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x109232: test1 (testcode.c:19)
==4510==    by 0x10955A: main (testcode.c:68)
==4510==  Block was alloc'd at
==4510==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x1091DE: test1 (testcode.c:11)
==4510==    by 0x10955A: main (testcode.c:68)
==4510==
Value of *ptr: 10
String: Good day to you!
==4510== Invalid write of size 4
==4510==    at 0x10931B: test3 (testcode.c:38)
==4510==    by 0x10956E: main (testcode.c:70)
==4510==  Address 0x4a98704 is 4 bytes inside a block of size 200 free'd
==4510==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x109328: test3 (testcode.c:39)
==4510==    by 0x10956E: main (testcode.c:70)
==4510==  Block was alloc'd at
==4510==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x1092DC: test3 (testcode.c:32)
==4510==    by 0x10956E: main (testcode.c:70)
==4510==
==4510== Invalid free() / delete / delete[] / realloc()
==4510==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x109328: test3 (testcode.c:39)
==4510==    by 0x10956E: main (testcode.c:70)
==4510==  Address 0x4a98700 is 0 bytes inside a block of size 200 free'd
==4510==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x109328: test3 (testcode.c:39)
==4510==    by 0x10956E: main (testcode.c:70)
==4510==  Block was alloc'd at
==4510==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4510==    by 0x1092DC: test3 (testcode.c:32)
==4510==    by 0x10956E: main (testcode.c:70)
==4510==
==4510== Invalid read of size 4
```

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 testcode.c"

Commands used:

gdb ./testcode

b test1, b test2, b test3, b test4 -> to add break points

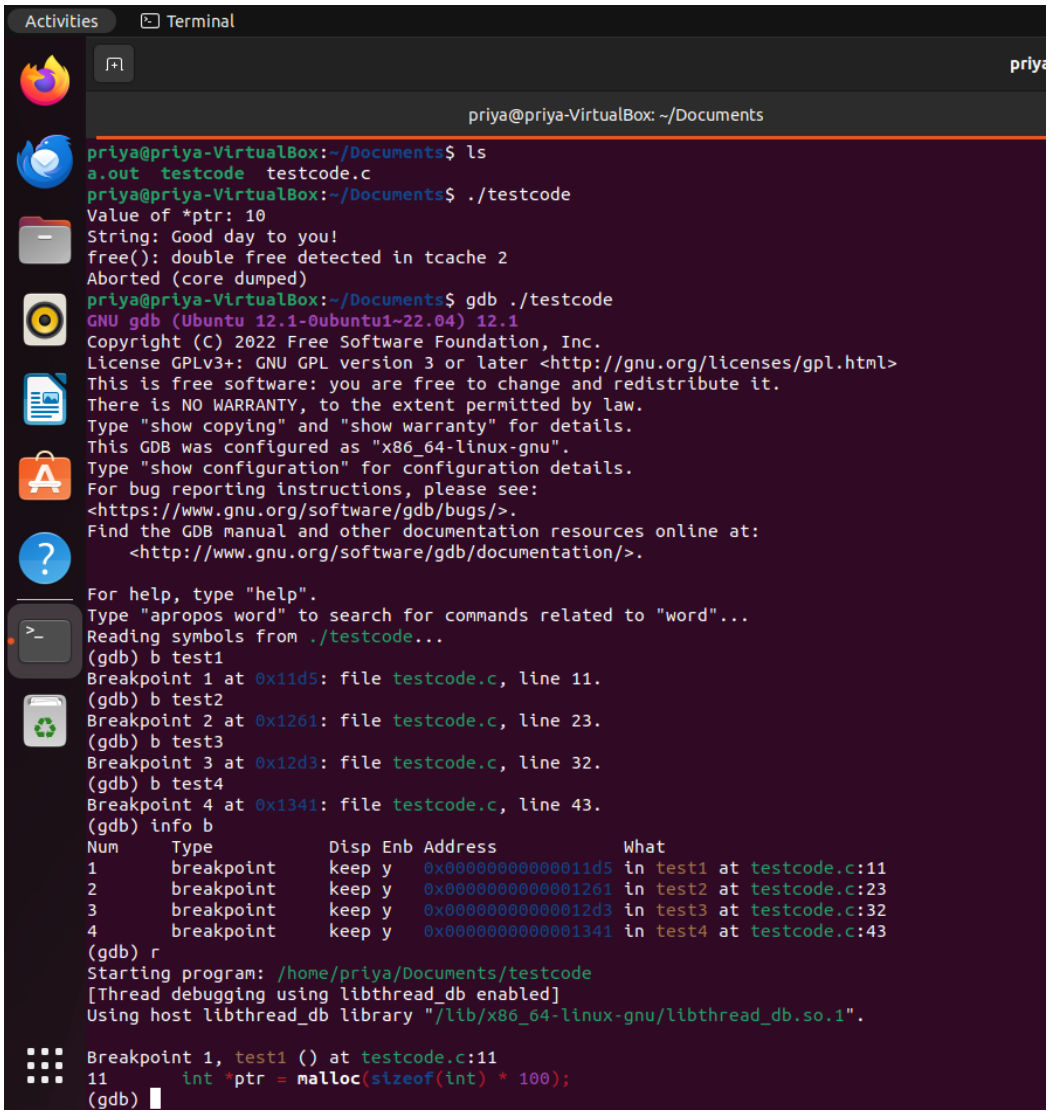
info b

r -> run the program with gdb

l -> listing the code

n -> next

c -> continuing



```
Breakpoint 1, test1 () at testcode.c:11
11        int *ptr = malloc(sizeof(int) * 100);
(gdb) l
6         char *name;
7         int id;
8         int *values;
9        } DataStruct;
10       void test1() {
11        int *ptr = malloc(sizeof(int) * 100);
12        if (ptr == NULL) {
13        perror("Failed to allocate memory");
14        return;
15        }
(gdb)
16        for (int i = 0; i < 100; i++) {
17        ptr[i] = i;
18        }
19        free(ptr);
20        printf("Value of *ptr: %d\n", ptr[10]);
21       }
22       void test2() {
23        char *str = malloc(100 * sizeof(char));
24        if (str == NULL) {
25        perror("Failed to allocate memory");
(gdb) n
12        if (ptr == NULL) {
(gdb) n
16        for (int i = 0; i < 100; i++) {
(gdb) n
17        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 testcode.c:23
23        char *str = malloc(100 * sizeof(char));
(gdb) c
Continuing.
```

Commands used:

disassemble -> to check the assembly code

disable break 2 -> to disable a specific breakpoint and continue

info registers -> to know about the value of registers

```
(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)
   0x00007ffff7c96962 <+146>:   cmp    $0x1,%edx
   0x00007ffff7c96965 <+149>:   jg     0x7ffff7c96a28 <__GI___pthread_kill+344>
   0x00007ffff7c9696b <+155>:   mov    $0x8,%r10d
   0x00007ffff7c96971 <+161>:   xor    %edx,%edx
   0x00007ffff7c96973 <+163>:   mov    %r14,%rsi
   0x00007ffff7c96976 <+166>:   mov    $0x2,%edi
--Type <RET> for more, q to quit, c to continue without paging--
```

```
1       breakpoint     keep y   0x00000000000011d5 in test1 at testcode.c:11
2       breakpoint     keep y   0x0000000000001261 in test2 at testcode.c:23
3       breakpoint     keep y   0x00000000000012d3 in test3 at testcode.c:32
(gdb) r
Starting program: /home/priya/Documents/testcode
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, test1 () at testcode.c:11
11        int *ptr = malloc(sizeof(int) * 100);
(gdb) disable break 2
(gdb) c
Continuing.
Value of *ptr: 10
String: Good day to you!

Breakpoint 3, test3 () at testcode.c:32
32        int *ptr = malloc(sizeof(int) * 50);
(gdb) info registers
rax            0x0                 0
rbx            0x0                 0
rcx            0x1                 1
rdx            0x0                 0
rsi            0x555555559440      93824992252992
rdi            0x7fffffffda70      140737488345712
rbp            0x7fffffffdfe0      0x7fffffffdfe0
rsp            0x7fffffffdfd0      0x7fffffffdfd0
r8             0x0                 0
r9             0x7ffff7d7c870      140737351501936
r10            0x0                 0
r11            0x246               582
r12            0x7fffffffe108      140737488347400
r13            0x555555555549      93824992236873
r14            0x555555557da0      93824992247200
r15            0x7ffff7ffd040      140737354125376
rip            0x5555555552d3      0x5555555552d3 <test3+12>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```