COMP 4736 Lab 2
Marco Ho Set T
A01338160

1. The program appears to be crashing on line 13 as it's trying to dereference a pointer that is pointing to NULL (0x0), which is an address space that the program is not allowed to access. This causes a "segmentation fault".

```
Program received signal SIGSEGV, Segmentation fault.
0x000055555555523e in main (argc=1, argv=0x7fffffffe0f8) at pointers.cpp:13
13          cout << *p << endl;
(gdb) list
8           int *q = NULL;
9
10          cout << *p << endl;
11
12          p = q;
13          cout << *p << endl;
14
15          p = &b;
16          cout << *p << endl;
17
(gdb)
```

*q stores NULL

assign NULL stored in q to p

Attempting to print NULL to standard output

2. The logical error would be due to the program updating the value of last prior to assigning it to second_last. This makes it so both last and second_last are

assigned the value stored in ‘next’.

```
1
1

Breakpoint 1, main (argc=1, argv=0x7fffffffe0a8) at fibonacci.cpp:13
13        for(int i=1; i<=10; i++) {
(gdb) n
14          int next = second_last + last;
(gdb) n
15          cout << next << endl;
(gdb) print second_last
$23 = 1
(gdb) print last
$24 = 1
(gdb) print next
$25 = 2
(gdb) n
2
16          last = next;
(gdb) n
17          second_last = last;
(gdb) n
13        for(int i=1; i<=10; i++) {
(gdb) n
14          int next = second_last + last;
(gdb) print next
$26 = 2
(gdb) print second_last
$27 = 2
(gdb) print last
$28 = 2
(gdb)
```

3.

```
==75100== ERROR SUMMARY: 6 errors from 6 contexts (suppressed: 0 from 0)
root@MHO-laptop:~# valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --num-callers=20 ./memory_bugs
==91760== Memcheck, a memory error detector
==91760== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==91760== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==91760== Command: ./memory_bugs
==91760==
==91760== Syscall param write(buf) points to uninitialised byte(s)
==91760==    at 0x4974887: write (write.c:26)
==91760==    by 0x109235: main (memory_bugs.c:19)
==91760==  Address 0x1ffefffec0 is on thread 1's stack
==91760==  in frame #1, created by main (memory_bugs.c:9)
==91760==
==91760== Invalid write of size 1
==91760==    at 0x109254: main (memory_bugs.c:26)
==91760==  Address 0x4a8c0a0 is 0 bytes inside a block of size 12 free'd
==91760==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x10924F: main (memory_bugs.c:23)
==91760==  Block was alloc'd at
==91760==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x10923F: main (memory_bugs.c:22)
==91760==
==91760== Invalid read of size 1
==91760==    at 0x10925B: main (memory_bugs.c:29)
==91760==  Address 0x4a8c0a0 is 0 bytes inside a block of size 12 free'd
==91760==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x10924F: main (memory_bugs.c:23)
==91760==  Block was alloc'd at
==91760==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x10923F: main (memory_bugs.c:22)
==91760==
A
==91760== Invalid free() / delete / delete[] / realloc()
==91760==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x109290: main (memory_bugs.c:35)
==91760==  Address 0x1ffefffec0 is on thread 1's stack
==91760==  in frame #1, created by main (memory_bugs.c:9)
==91760==
==91760==
==91760== HEAP SUMMARY:
==91760==     in use at exit: 80 bytes in 2 blocks
==91760==   total heap usage: 4 allocs, 3 frees, 1,116 bytes allocated
==91760==
==91760== 30 bytes in 1 blocks are definitely lost in loss record 1 of 2
==91760==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x10920D: main (memory_bugs.c:16)
==91760==
==91760== 50 bytes in 1 blocks are definitely lost in loss record 2 of 2
==91760==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==91760==    by 0x109280: main (memory_bugs.c:32)
==91760==
==91760== LEAK SUMMARY:
==91760==    definitely lost: 80 bytes in 2 blocks
==91760==    indirectly lost: 0 bytes in 0 blocks
==91760==      possibly lost: 0 bytes in 0 blocks
==91760==    still reachable: 0 bytes in 0 blocks
==91760==         suppressed: 0 bytes in 0 blocks
==91760==
==91760== Use --track-origins=yes to see where uninitialised values come from
==91760== For lists of detected and suppressed errors, rerun with: -s
==91760== ERROR SUMMARY: 6 errors from 6 contexts (suppressed: 0 from 0)
```

Errors:

1. Error: “Syscall param write(buf) points to uninitialized byte(s)”.
   Explanation: The array being passed to write() in line 19 (memory_bugs.c:19) has been declared by not defined; the array currently contains 10 nulls at the time of passing it to the write function and printing null(s) in C causes an error.

2. Error: “Invalid write of size 1”
   Explanation: This error occurred at line 26 (memory_bugs.c:26) because the program is trying to store character ‘A’ at the memory space where pointer *P* is pointing to despite after having ‘freed’ (released) the memory space on line 23.
3. Error: “Invalid read of size 1”
   Explanation: Like error 2, on line 29 (memory_bugs.c:29), the program is trying access memory space that has already been released.
4. Error: “Invalid free() / delete / delete[] / realloc()”
   Explanation: On line 35 (memory_bugs.c:35), the program is invoking free() on arr which is an int array of size 10 that was declared but never defined nor was it ever allocated with malloc/calloc/realloc.
5. Error: “30 bytes in 1 blocks are definitely lost in loss record 1 of 2”
   Explanation: 30 bytes were allocated with malloc and assigned to *p* on line 16 (memory_bugs.c:16) but then allocates 12 different bytes to the same variable *p* on line 22 (memory_bugs.c:22) without invoking free() and releasing the 30 initial bytes.
6. Error: “50 bytes in 1 blocks are definitely lost in loss record 2 of 2”
   Explanation: 50 bytes were allocated and the address of the 50 bytes were stored in pointer *q* in line 32 (memory_bugs.c:32) but were never free’d/released.