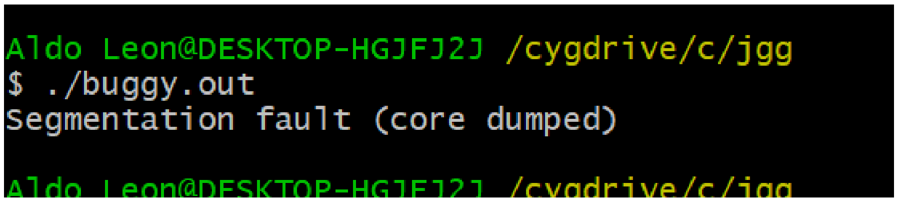
CSCE 313 Programming Assignment 0

Aldo Leon Marquez UIN: 326004699

The Purpose of the assignment was to recap some concepts of c++ and to introduce us to setup and use a linux environment to compile and debug the given code and for future assignments during the course. The assignment consists of a code that had some bugs that we were to find and solve.

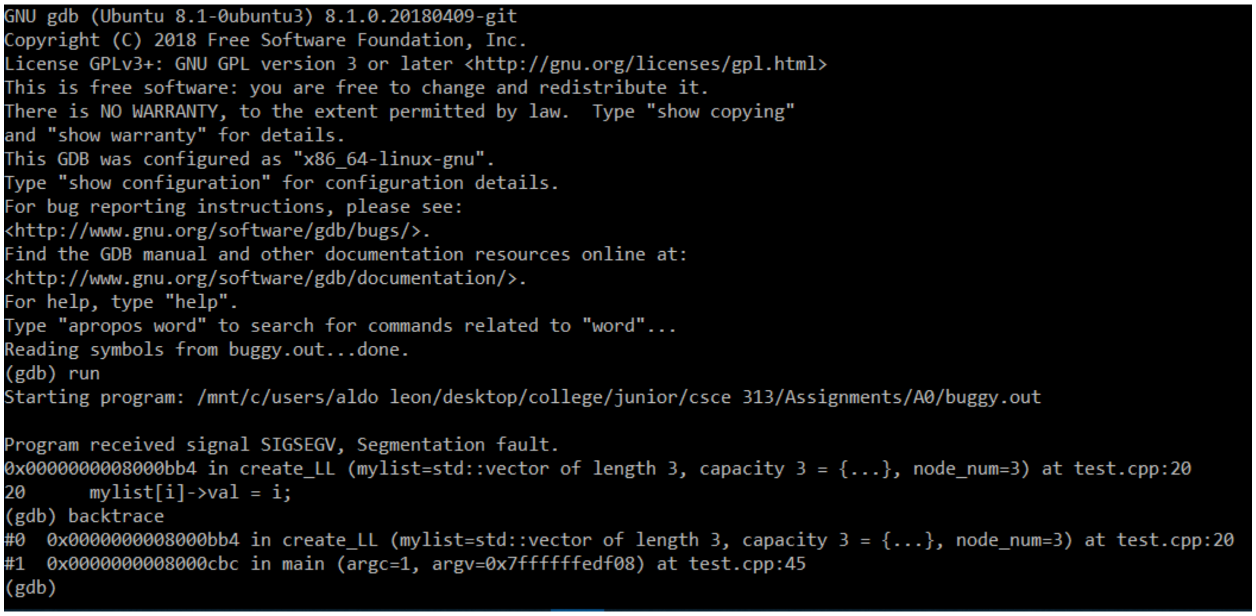
(**4)** After solving the first few exercises, we encounter a segmentation fault when first trying to run the .out file:



```
Aldo Leon@DESKTOP-HGJFJ2J /cygdrive/c/jgg
$ ./buggy.out
Segmentation fault (core dumped)

Aldo Leon@DESKTOP-HGJFJ2J /cygdrive/c/jgg
```

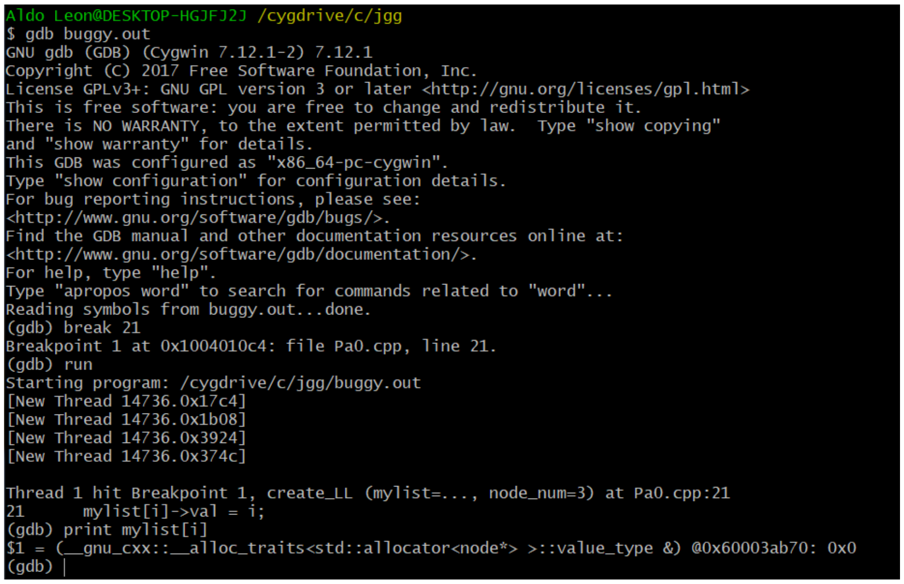
(**6**) Then the gdb is used to start solving the bugs that prevent the program from running:



```
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 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:
<http://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 buggy.out...done.
(gdb) run
Starting program: /mnt/c/users/aldo leon/desktop/college/junior/csce 313/Assignments/A0/buggy.out

Program received signal SIGSEGV, Segmentation fault.
0x0000000008000bb4 in create_LL (mylist=std::vector of length 3, capacity 3 = {...}, node_num=3) at test.cpp:20
20      mylist[i]->val = i;
(gdb) backtrace
#0  0x0000000008000bb4 in create_LL (mylist=std::vector of length 3, capacity 3 = {...}, node_num=3) at test.cpp:20
#1  0x0000000008000cbc in main (argc=1, argv=0x7ffffffedf08) at test.cpp:45
(gdb)
```

(**7**) Now that we know in what line the error is. By printing the stored value the issue becomes clear, the pointer stores in the vector is pointing to address 0x0;

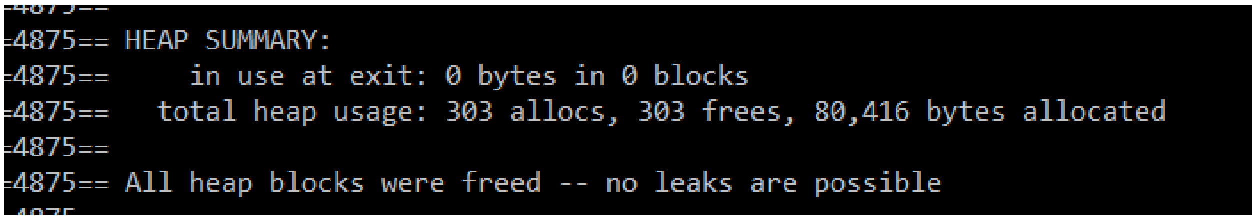


```
Aldo Leon@DESKTOP-HGJFJ2J /cygdrive/c/jgg
$ gdb buggy.out
GNU gdb (GDB) (Cygwin 7.12.1-2) 7.12.1
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-pc-cygwin".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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 buggy.out...done.
(gdb) break 21
Breakpoint 1 at 0x1004010c4: file Pa0.cpp, line 21.
(gdb) run
Starting program: /cygdrive/c/jgg/buggy.out
[New Thread 14736.0x17c4]
[New Thread 14736.0x1b08]
[New Thread 14736.0x3924]
[New Thread 14736.0x374c]

Thread 1 hit Breakpoint 1, create_LL (mylist=..., node_num=3) at Pa0.cpp:21
21      mylist[i]->val = i;
(gdb) print mylist[i]
$1 = (__gnu_cxx::__alloc_traits<std::allocator<node*> >::value_type &) @0x60003ab70: 0x0
(gdb) |
```

(**9**) The lst segmentation fault we encounter has to do with how the for loop is setup when creating the node list, the loop runs one to many times so it tries to access an out of index pointer, causing the segfault.

(**10**) The make sure the nodes were deleted and no memory leaks occur I implemented a for loop that runs for the final size of the vector, assigning the “next” pointers to Null and then erasing the pointer itself from the vector. A valgrind check for memory leaks confirms that no leak occurred.



```
=4875== HEAP SUMMARY:
=4875==     in use at exit: 0 bytes in 0 blocks
=4875==   total heap usage: 303 allocs, 303 frees, 80,416 bytes allocated
=4875==
=4875== All heap blocks were freed -- no leaks are possible
```