Task 1: Programming Tools

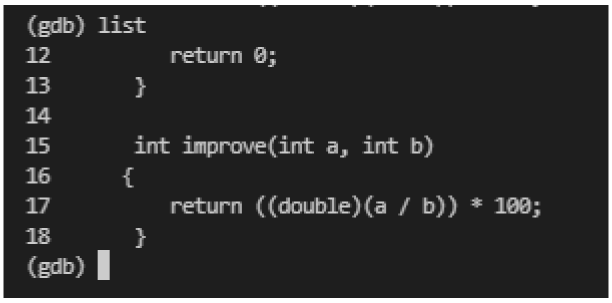
1.

1.1.1. Command run *g++ -g marks.cpp, gdb a.out*

1.1.2. Compiled and run through gdb, used command *run.* Issue found: line 17 when arguments a = -2, b = 0 and breaks on line 18 (a/b).

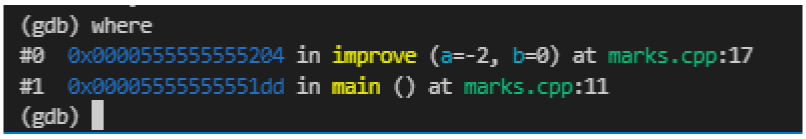
1.1.3. Command run *gdb a.out*

1.1.4. Command run *where*



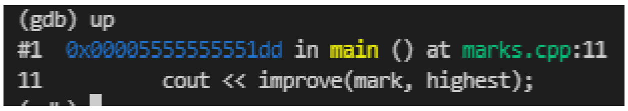
```
(gdb) list
12          return 0;
13      }
14
15      int improve(int a, int b)
16     {
17          return ((double)(a / b)) * 100;
18      }
(gdb)
```

1.1.5.



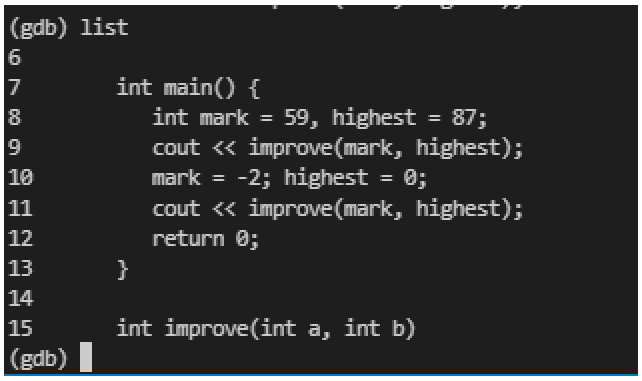
```
(gdb) where
#0  0x0000555555555204 in improve (a=-2, b=0) at marks.cpp:17
#1  0x00005555555551dd in main () at marks.cpp:11
(gdb)
```

1.1.6.



```
(gdb) up
#1  0x00005555555551dd in main () at marks.cpp:11
11          cout << improve(mark, highest);
```

1.1.7. Command run *up*



```
(gdb) list
6
7       int main() {
8           int mark = 59, highest = 87;
9           cout << improve(mark, highest);
10          mark = -2; highest = 0;
11          cout << improve(mark, highest);
12          return 0;
13      }
14
15      int improve(int a, int b)
(gdb)
```

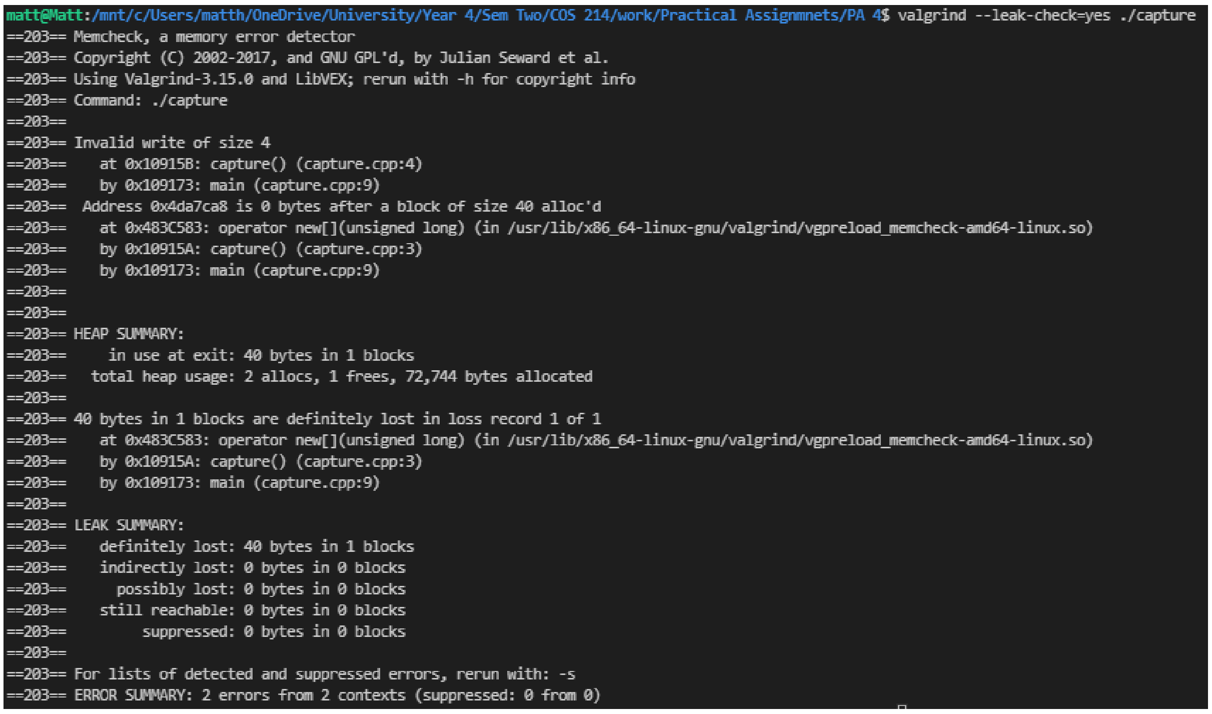
1.1.8. Command run *print highest*

1.1.9. The problem that was found is he variable highest was set to 0 and used in the division mark divided by 0.

1.2.

1.2.1. Commands run: *g++ -g -Og capture.cpp -o capture*

1.2.2. Command run *valgrind –leak-check=yes ./capture*



```
matt@Matt:/mnt/c/Users/matth/OneDrive/University/Year 4/Sem Two/COS 214/work/Practical Assignmnets/PA 4$ valgrind --leak-check=yes ./capture
==203== Memcheck, a memory error detector
==203== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==203== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==203== Command: ./capture
==203==
==203== Invalid write of size 4
==203==    at 0x10915B: capture() (capture.cpp:4)
==203==    by 0x109173: main (capture.cpp:9)
==203==  Address 0x4da7ca8 is 0 bytes after a block of size 40 alloc'd
==203==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==203==    by 0x10915A: capture() (capture.cpp:3)
==203==    by 0x109173: main (capture.cpp:9)
==203==
==203==
==203== HEAP SUMMARY:
==203==     in use at exit: 40 bytes in 1 blocks
==203==   total heap usage: 2 allocs, 1 frees, 72,744 bytes allocated
==203==
==203== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==203==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==203==    by 0x10915A: capture() (capture.cpp:3)
==203==    by 0x109173: main (capture.cpp:9)
==203==
==203== LEAK SUMMARY:
==203==    definitely lost: 40 bytes in 1 blocks
==203==    indirectly lost: 0 bytes in 0 blocks
==203==      possibly lost: 0 bytes in 0 blocks
==203==    still reachable: 0 bytes in 0 blocks
==203==         suppressed: 0 bytes in 0 blocks
==203==
==203== For lists of detected and suppressed errors, rerun with: -s
==203== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

1.2.3. 203

1.2.4. 40 Bytes have definitely been lost.

1.2.5. The error has occurred in line 3 where the *new* keyword has been used and this was called from line 9.

1.2.6. 40 Bytes has been lost because memory was allocated for the int array but not deallocated.

1.2.7. In capture() I'd add before the function scope ends *delete marks;* to deallocate the memory.