

---

## HW10: Valgrind

---

### Part A

1. Given the following output from a program running under Valgrind

```
8 bytes in 1 blocks are definitely lost in loss record 1 of 7
   at 0x489C344: operator new[](unsigned long) (vg_replace_malloc.c:433)
   by 0x1091CB: main (valgrind-test.cpp:5)
```

What is the meaning of the output?

A program does not free the allocated memory.

2. Given the following output from a program running under Valgrind

```
Invalid write of size 8
   at 0x10942F: new_teams(int, Team**) (program_2.cpp:30)
   by 0x109478: main (program_2.cpp:35)
Address 0x0 is not stack'd, malloc'd or (recently) free'd
```

What problem does this output indicate?

The program is trying to write to a memory space it does not own.

3. Given the following two commands, is the Valgrind command correct?  
If not, what is the reason that makes the command incorrect?

```
g++ -Wall -O0 program_2.cpp -o program_2
valgrind ./program_2.cpp
```

Not correct because Valgrind is called on a source code and not an executable file.

4. Given the following output from a program running under Valgrind

```
==2404== Use of uninitialised value of size 8
==2404==    at 0x497910B: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==2404==    by 0x497942C: std::ostreambuf_iterator<char, std::char_traits<char> > std::num_put<char,
std::ostreambuf_iterator<char, std::char_traits<char> > >::_M_insert_int<long>(std::ostreambuf_iterator<char,
std::char_traits<char> >, std::ios_base&, char, long) const (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==2404==    by 0x4987D5E: std::ostream& std::ostream::_M_insert<long>(long) (in
/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==2404==    by 0x109214: main (program_1.cpp:20)
```

What does the number 2404 in ==2404== mean?

The process ID of the program being run under Valgrind.

5. Given the following output from a program running under Valgrind

```
==7638== HEAP SUMMARY:
==7638==    in use at exit: 120 bytes in 1 blocks
==7638==    total heap usage: 2 allocs, 1 frees, 72,824 bytes allocated
==7638==
==7638== LEAK SUMMARY:
==7638==    definitely lost: 120 bytes in 1 blocks
==7638==    indirectly lost: 0 bytes in 0 blocks
==7638==    possibly lost: 0 bytes in 0 blocks
==7638==    still reachable: 0 bytes in 0 blocks
==7638==    suppressed: 32 bytes in 0 blocks
==7638== Rerun with --leak-check=full to see details of leaked memory
```

Is there a memory loss, if so how many bytes are lost?

120 bytes are lost.

---

## Part B

For each of the programs in the `part-b` folder, show

1. The Valgrind command you used to analyze the program.
2. The problem found by Valgrind. You may use a screenshot of the output, but highlight the problem you want to talk about.
3. Description of the fix you apply.
4. Link to the commit that you fixed the problem. The fixed source code should be committed and pushed to GitHub.

	test1.cpp	test2.cpp	test3.cpp
Valgrind command	valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes -s ./test2	valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes -s ./test3	valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes -s ./test4
output screenshot			
problem	Conditional jump or move depends on uninitialised value(s)  (uninitialized variable)	Access not within mapped region at address 0x0  (the program is trying to write to a memory space it does not own)	Invalid free() / delete / delete[] / realloc()  (memory loss/leakage due to trying to free the same block of memory twice)
fix description	set <code>bool should_log_user_in = true;</code> so that <code>should_log_user_in</code> is initialized	comment out the line <code>x[0] = 10;</code> will get an unused-but-set-variable warning	second one should be <code>free(acc2);</code> so that both accounts are being freed
commit link	<a href="https://github.com/OU-CS3560/hw10-valgrind-s22-23-hc219417/commit/8d2df4107f30211f3f5c2285a75b1769fda16cfe">https://github.com/OU-CS3560/hw10-valgrind-s22-23-hc219417/commit/8d2df4107f30211f3f5c2285a75b1769fda16cfe</a>		

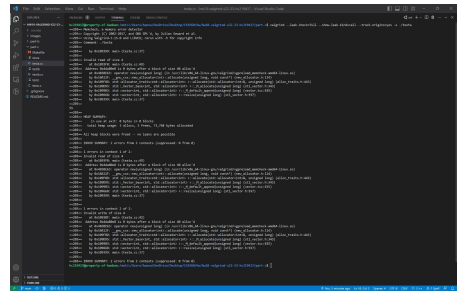
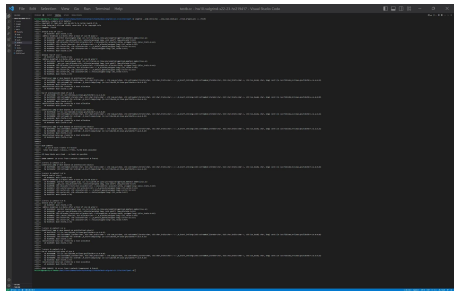
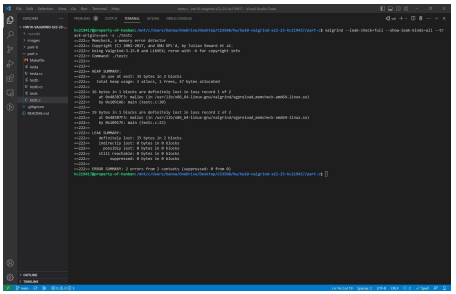
## Part C

For each of the three programs you selected

1. Explain which program you selected.

test programs (source code files) are from valgrind examples folder on [github](#) (ppt lecture)  
(had to rename `test.cc` to `testb.cc` and `test.c` to `testc.c` for compilation purposes)

2. What action did you perform while running the program under Valgrind?
3. List the problem(s) you found.
4. Answer the following two questions:
  - a. How much memory (in bytes) is leaked that is not managed by the program?
  - b. How much memory allocation the program performed?

	testa.cc	testb.cc	testc.c
action performed	compiled using Makefile, then by running the following command: <pre>valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes -s ./testa</pre>	compiled using Makefile, then by running the following command: <pre>valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes -s ./testb</pre>	compiled using Makefile, then by running the following command: <pre>valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes -s ./testc</pre>
screenshot			

<b>problem(s)</b>	<ul style="list-style-type: none"> <li>Invalid write of size 4</li> <li>Invalid read of size 4</li> </ul> <p>(2 errors from 2 contexts)</p>	<ul style="list-style-type: none"> <li>Invalid write of size 4 (<b>x2</b>)</li> <li>Conditional jump or move depends on uninitialised value(s) (<b>x9</b>)</li> <li>Use of uninitialised value of size 8 (<b>x7</b>)</li> </ul> <p>(18 errors from 6 contexts)</p>	<ul style="list-style-type: none"> <li>16 bytes in 1 blocks are definitely lost in loss record 1 of 2</li> <li>19 bytes in 1 blocks are definitely lost in loss record 2 of 2</li> </ul> <p>(2 errors from 2 contexts)</p>
<b>memory leaked</b>	0 bytes		35 bytes
<b>memory allocated</b>	73,768 bytes		47 bytes

Write out your reflection on the usefulness of Valgrind.

Valgrind is a highly useful tool because it captures problems other tools may have missed. Valgrind checks the internals of a program as opposed to comparing output with expected output. It also detects memory leaks and memory errors which is super important when it comes to coding!! :O

---