

1. I checked curl on WSL by using: `valgrind --leak-check=full --show-leak-kinds=all curl`  
This did not find any memory leaks of any sort with the curl command, and the same thing happened when checking curl <https://www.cnn.com>, but above the summary section of valgrind a lot of output was shown in powershell.

```
hugh@hugh-PC: ~$ valgrind --leak-check=full --show-leak-kinds=all curl
==551== Memcheck, a memory error detector
==551== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==551== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==551== Command: curl
==551==
==551== error calling PR_SET_PTRACER, vgdb might block
curl: try 'curl --help' or 'curl --manual' for more information
==551==
==551== HEAP SUMMARY:
==551==   in use at exit: 0 bytes in 0 blocks
==551==   total heap usage: 4,609 allocs, 4,609 frees, 273,220 bytes allocated
==551==
==551== All heap blocks were freed -- no leaks are possible
==551==
==551== For counts of detected and suppressed errors, rerun with: -v
==551== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
hugh@hugh-PC: ~$
```

```
"ads-person","content-person-prof","content-person","measure-ads","measure-content","measure-market","product-develop"]
));(document);</script><script>(function($){$(document).onZonesAndDomReady(function () {if (Modernizr.android === true)
{ $('head').append('<meta name="theme-color" content="#000000">');} $('head').append('<link href="//cdn.cnn.com/cnn/e/img
/3.0/global/misc/cnn-logo.png" rel="icon" sizes="192x192">');});})(jQuery);</script><script data-obct>!function(_window
, _document) {var OB_ADV_ID = '00b2d6c56fc76084821b9d05abf2f201d9',api;if (_window.obApi) {if (!Array.isArray(_window.ob
Api.marketerId)) {_window.obApi.marketerId = [_window.obApi.marketerId];}return;}api = _window.obApi = function () {var
a = window.obApi;a.dispatch ? a.dispatch.apply(a, arguments) : a.queue.push(arguments);};api.version = '1.0';api.loaded
= true;api.marketerId = OB_ADV_ID;api.queue = [];WM.UserConsent.addScript({async: true, src: '//amplify.outbrain.com/cp/
obtp.js'}, [{"iab","vendor","data-store","ads-contextual","ads-person-prof","ads-person","content-person-prof","content-p
erson","measure-ads","measure-content","measure-market","product-develop"}, _document.getElementsByTagName('script')[0].
parentNode]);}(window, document);obApi('track', 'PAGE_VIEW');document.addEventListener('click', function(event) {var el =
event.target;if (el && (el.tagName === 'A' || (el.parentNode && el.parentNode.tagName === 'A')))) {obApi('track', 'Exit
Link Clicks');});</script><script>if (WM.UserConsent.inUserConsentState(["vendor","measure-content"], { id: "twitterPix
el" })) {(function (w, d) {var e,s;if (!w.twq) {s = w.twq = function () {s.exe ? s.exe.apply(s, arguments) : s.queue.pus
h(arguments);};s.version = '1.1';s.queue = [];e = d.createElement('script');e.async = true;e.type = 'text/javascript';e.
src = '//static.ads-twitter.com/uwt.js';d.getElementsByTagName('body')[0].appendChild(e);}})(window, document);twq('init
', 'nyutn');twq('track', 'PageView');</script><script>WM.UserConsent.addScript({async: true,defer: true,src: "//get.s-o
netag.com/c15dde9-ec7d-4a49-b8ca-7a21bc4b943b/tag.min.js"}, [{"iab","vendor","data-store","ads-contextual","ads-person-p
rof","ads-person","content-person-prof","content-person","measure-ads","measure-content","measure-market","product-devel
op"}]);</script><script>WM.UserConsent.addScript({async: true,src: "//tru.am/scripts/custom/cnn.js"}, [{"data-store","ads-
contextual","ads-person-prof","ads-person"}]);</script><script>(function (win, doc, WM) {win['bt'] = win['bt'] || functio
n () { (win['_bt'] = win['_bt'] || []).push(arguments); };WM.UserConsent.addScript({async: true,src: '//cdn.boomtrain.co
m/pl13n/cnn/pl13n.min.js'}, [{"vendor","data-store","ads-contextual","ads-person-prof","ads-person","content-person-prof","
content-person","measure-ads","measure-content","measure-market","product-develop"}]);(window, document, WM);</script><
div class="ad ad--epic ad--all-skin"><div data-ad-id="ad_oop_skin_01" id="ad_oop_skin_01" class="ad-ad_oop_skin_01 ad-re
fresh-adbanner"></div></div><div class="ad ad--epic ad--all"><div data-ad-id="ad_oop_float_01" id="ad_oop_float_01" clas
s="ad-ad_oop_float_01 ad-refresh-adbody"></div></div></body></html>==549==
==549== HEAP SUMMARY:
==549==   in use at exit: 0 bytes in 0 blocks
==549==   total heap usage: 35,527 allocs, 35,527 frees, 5,415,320 bytes allocated
==549==
==549== All heap blocks were freed -- no leaks are possible
==549==
==549== For counts of detected and suppressed errors, rerun with: -v
==549== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
hugh@hugh-PC: ~$
```

Using Valgrind on the word count command, it seems the program has no issues at all with memory and output the correct numbers.

```
hugh@hugh-PC: ~$ valgrind --leak-check=full --show-leak-kinds=all wc index.html
]==554== Memcheck, a memory error detector
]==554== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
]==554== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
]==554== Command: wc index.html
]==554==
]==554== error calling PR_SET_PTRACER, vgdb might block
183  30879 1131411 index.html
]==554==
]==554== HEAP SUMMARY:
]==554==   in use at exit: 0 bytes in 0 blocks
]==554==   total heap usage: 233 allocs, 233 frees, 18,827 bytes allocated
]==554==
]==554== All heap blocks were freed -- no leaks are possible
]==554==
]==554== For counts of detected and suppressed errors, rerun with: -v
]==554== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
hugh@hugh-PC: ~$ ]
```

6)

Using cppcheck on program.cc, it seems that no matter what I tried cppcheck could't find any errors, but I received the error that cppcheck could not find the include files.

```
hugh@hugh-PC: ~$ cppcheck --enable=all program.cc
Checking program.cc ...
(information) Cppcheck cannot find all the include files (use --check-config for details)
```

Using valgrind on the program, there were no errors or memory leaks found.

```
hugh@hugh-PC: ~$ valgrind ./count
==1484== Memcheck, a memory error detector
==1484== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1484== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==1484== Command: ./count
==1484==
==1484== error calling PR_SET_PTRACER, vgdb might block
1 Lines.
6 Characters.
==1484==
==1484== HEAP SUMMARY:
==1484==   in use at exit: 0 bytes in 0 blocks
==1484==   total heap usage: 2 allocs, 2 frees, 73,216 bytes allocated
==1484==
==1484== All heap blocks were freed -- no leaks are possible
==1484==
==1484== For counts of detected and suppressed errors, rerun with: -v
==1484== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

7 and 8)

Unfortunately I could not figure out what was wrong with my test file or include paths as I was getting the errors resulting below, I did not have enough time to figure them out, sorry.

```
hugh@hugh-PC: ~/lib$ make test
g++ -Wall -c program.cc
g++ -Wall -I./lib -c -o test.o test.cc
In file included from test.cc:3:0:
program.cc: In function 'int main(int, char**)':
program.cc:38:5: error: redefinition of 'int main(int, char*)'
  int main(int argc, char *argv[])
      ^
In file included from test.cc:2:0:
catch.hpp:17441:5: note: 'int main(int, char*)' previously defined here
  int main (int argc, char * argv[]) {
      ^
<builtin>: recipe for target 'test.o' failed
make: *** [test.o] Error 1
```