Homework 1 Report 2

Check assign pointers

Assign 100 ints

```
porenp@burt-reynolds 12:24:18 ~/compsci250p/hw1
$  vim valgrind.cpp
porenp@burt-reynolds 12:24:34 ~/compsci250p/hw1
$ g++ valgrind.cpp
porenp@burt-reynolds 12:24:36 ~/compsci250p/hw1
$ valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --track-origins=yes a.out
==16448== Memcheck, a memory error detector
==16448== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==16448== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==16448== Command: a.out
==16448==
==16448==
==16448== HEAP SUMMARY:
==16448==     in use at exit: 73,104 bytes in 2 blocks
==16448==   total heap usage: 2 allocs, 0 frees, 73,104 bytes allocated
==16448==
==16448== 400 bytes in 1 blocks are definitely lost in loss record 1 of 2
==16448==    at 0x4C28BE3: malloc (vg_replace_malloc.c:299)
==16448==    by 0x40061A: main (in /home/porenp/compsci250p/hw1/a.out)
==16448==
==16448== 72,704 bytes in 1 blocks are still reachable in loss record 2 of 2
==16448==    at 0x4C28BE3: malloc (vg_replace_malloc.c:299)
==16448==    by 0x4EBD18F: pool (eh_alloc.cc:117)
==16448==    by 0x4EBD18F: __static_initialization_and_destruction_0 (eh_alloc.cc:244)
==16448==    by 0x4EBD18F: _GLOBAL__sub_I_eh_alloc.cc (eh_alloc.cc:307)
==16448==    by 0x400F552: _dl_init (in /usr/lib64/ld-2.17.so)
==16448==    by 0x40011A9: ??? (in /usr/lib64/ld-2.17.so)
==16448==
==16448== LEAK SUMMARY:
==16448==    definitely lost: 400 bytes in 1 blocks
==16448==    indirectly lost: 0 bytes in 0 blocks
==16448==      possibly lost: 0 bytes in 0 blocks
==16448==    still reachable: 72,704 bytes in 1 blocks
==16448==         suppressed: 0 bytes in 0 blocks
==16448==
==16448== For counts of detected and suppressed errors, rerun with: -v
==16448== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
porenp@burt-reynolds 12:25:06 ~/compsci250p/hw1
$
```

Figure 1: Valgrind result for assigning 100 ints without free memory

Assign 1000 doubles

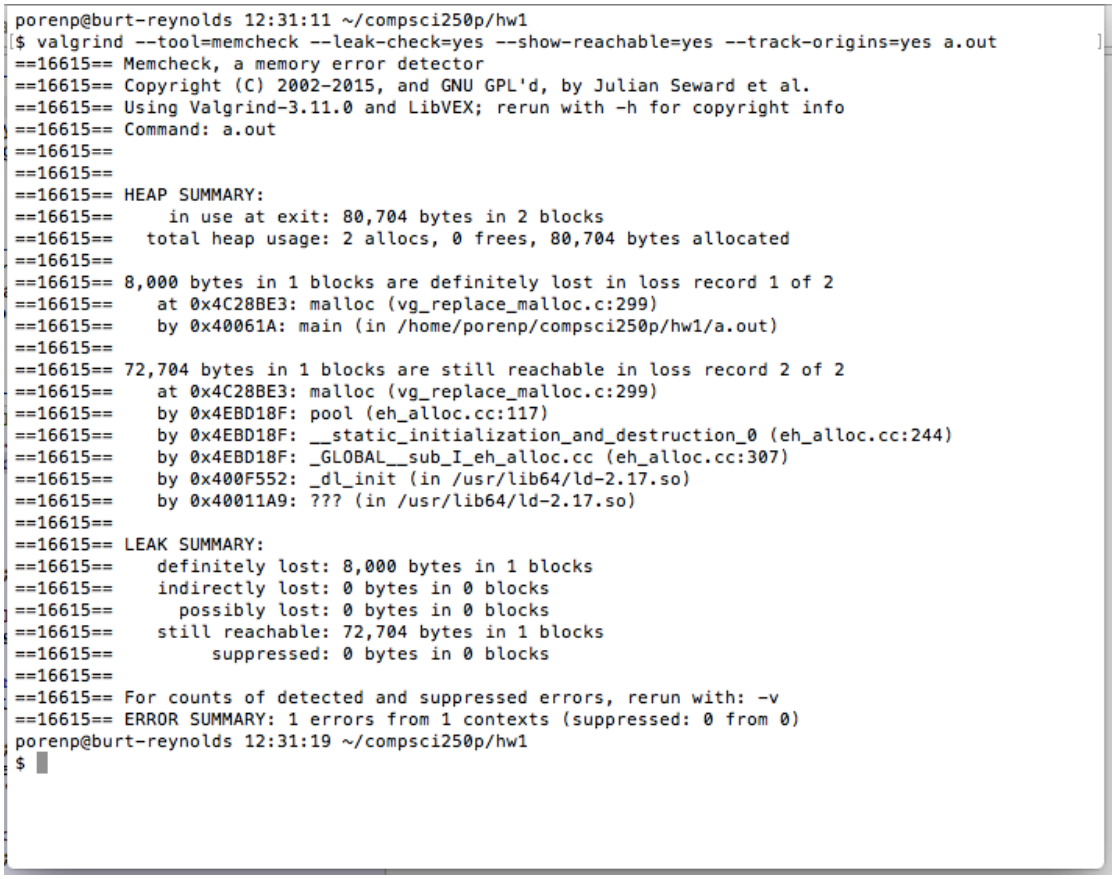

Figure 2: Valgrind result for assigning 1000 doubles without free memory

Questions

- Do you see any memory leak in either or both scenarios?
  Yes. Both scenarios have memory leak.
- How much memory leak did you see?
  400 Bytes for 100 ints
  8000 Bytes for 1000 doubles
- Does the memory leak match with your allocations ?
  Yes. Each int is 4 bytes and each double is 8 bytes. Then, multiply by 100 and 1000. We will get 400 bytes and 8000 bytes for those cases.

Check segmentation fault

Scenario 1

Malloc 10 ints and assign 1000 ints to the array

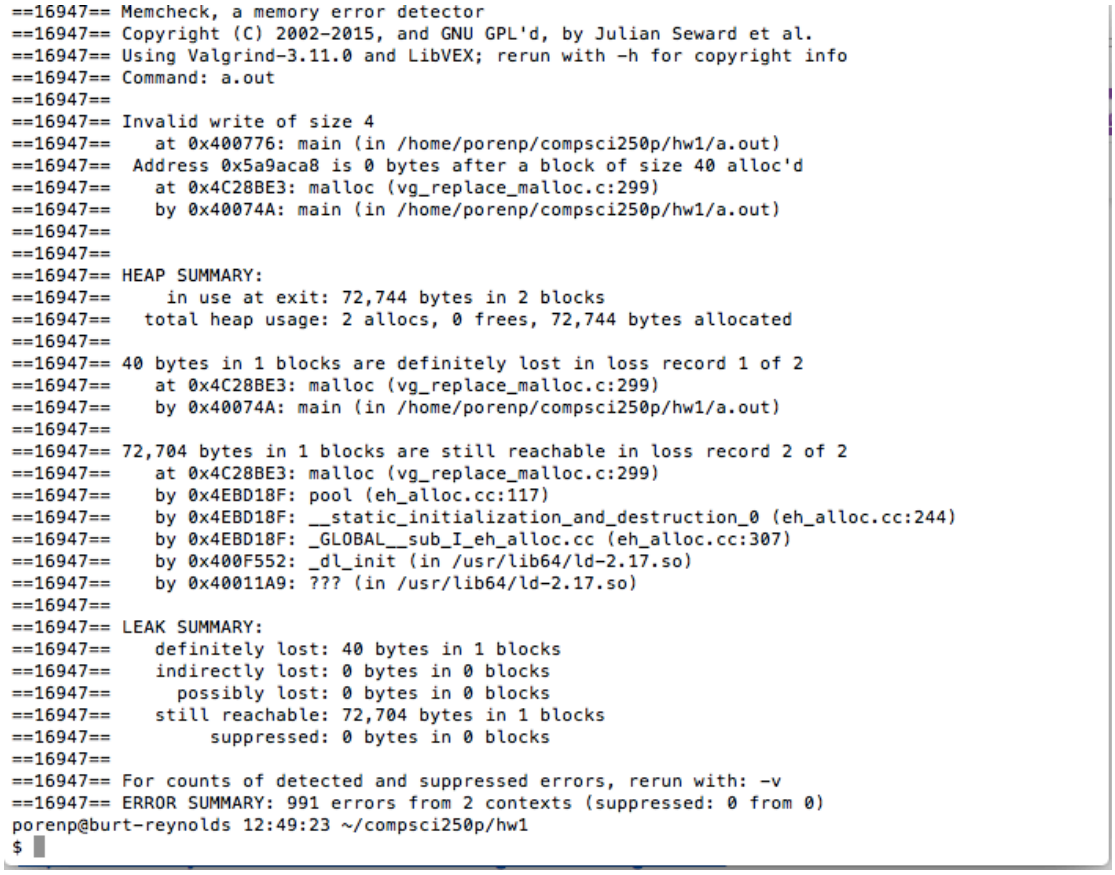

Figure 3: Valgrind result for allocating 10 ints and assigning 1,000 ints to this array

Questions

- What is the valgrind output?
  Because this scenario still doesn't free the memory so there are 40 bytes in block 1 loss. Besides, this time should have invalid write of size 4 because it writes outside the int_arr's allocation.
- Can you run the program without any errors?
  Yes. That's because the program doesn't write the memory block that has been used yet.
- Try to increase the size of the iteration until you get a segfault. What happens when you run it independently and with valgrind?
  This time I increase the loop til 100,000. Then, run the program independently, and get the segmentation fault.

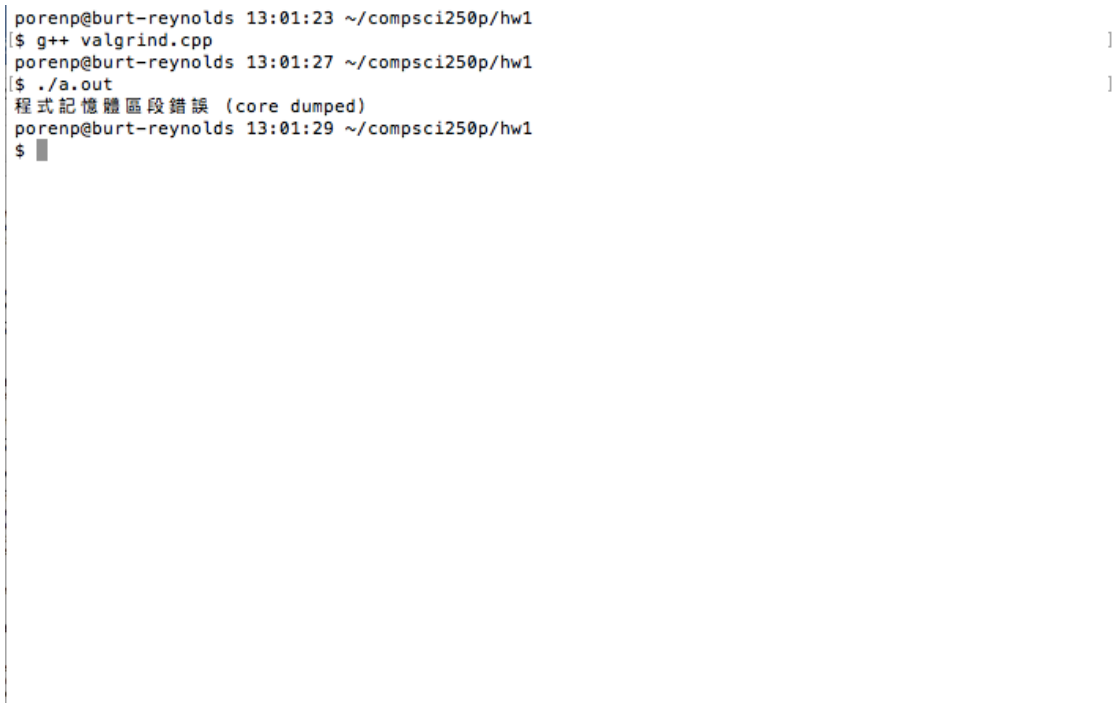

Figure 3: Execution result for allocating 10 ints and assigning 100,000 ints to this array

Now, run the program using Valgrind.

```
porenp@andromeda-5 21:18:41 ~/compsci250p
$ ls
.git/  hw0/  hw1/
porenp@andromeda-5 21:18:42 ~/compsci250p
$ cd hw1
porenp@andromeda-5 21:18:44 ~/compsci250p/hw1
$ ls
a.out*  dict.txt  prof.cpp  valgrind.cpp
porenp@andromeda-5 21:18:44 ~/compsci250p/hw1
$ valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --track-origins=yes a.out
==8170== Memcheck, a memory error detector
==8170== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==8170== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==8170== Command: a.out
==8170==
==8170== Invalid write of size 4
==8170==    at 0x4007C6: main (in /home/porenp/compsci250p/hw1/a.out)
==8170==  Address 0x5a9aca8 is 0 bytes after a block of size 40 alloc'd
==8170==    at 0x4C28BE3: malloc (vg_replace_malloc.c:299)
==8170==    by 0x40079A: main (in /home/porenp/compsci250p/hw1/a.out)
==8170==
==8170==
==8170== HEAP SUMMARY:
==8170==     in use at exit: 72,704 bytes in 1 blocks
==8170==   total heap usage: 2 allocs, 1 frees, 72,744 bytes allocated
==8170==
==8170== 72,704 bytes in 1 blocks are still reachable in loss record 1 of 1
==8170==    at 0x4C28BE3: malloc (vg_replace_malloc.c:299)
==8170==    by 0x4EBD18F: pool (eh_alloc.cc:117)
==8170==    by 0x4EBD18F: __static_initialization_and_destruction_0 (eh_alloc.cc:244)
==8170==    by 0x4EBD18F: _GLOBAL__sub_I_eh_alloc.cc (eh_alloc.cc:307)
==8170==    by 0x400F552: _dl_init (in /usr/lib64/ld-2.17.so)
==8170==    by 0x40011A9: ??? (in /usr/lib64/ld-2.17.so)
==8170==
==8170== LEAK SUMMARY:
==8170==    definitely lost: 0 bytes in 0 blocks
==8170==    indirectly lost: 0 bytes in 0 blocks
==8170==      possibly lost: 0 bytes in 0 blocks
==8170==    still reachable: 72,704 bytes in 1 blocks
==8170==         suppressed: 0 bytes in 0 blocks
==8170==
==8170== For counts of detected and suppressed errors, rerun with: -v
==8170== ERROR SUMMARY: 99990 errors from 1 contexts (suppressed: 0 from 0)
porenp@andromeda-5 21:19:44 ~/compsci250p/hw1
$
```

Figure 4: Valgrind result for allocating 10 ints and assigning 100,000 ints to this array

It shows more ERROR SUMMARY in the bottom.