# Ch.14 Coding Homework

Tyler Holmquist

gcc (Ubuntu 13.2.0-4ubuntu3) 13.2.0

1. When I ran this program it gave me an error stating “Segmentation fault (core dumped)” (gcc -g -o null null.c)
2. Gdb told me that there was a segmentation fault in main from the following error message, “Program received signal SIGSEGV, Segmentation fault. 0x0000aaaaaaaa076c in main ()”
3. When running with valgrind it tells us that there was a segmentation fault and that we had an invalid read of size four for 0x0 which is our null int. It also tells us that the program was terminated due to a SIGSEGV error which indicates a segmentation fault.
4. When the program runs it completes as normal with no errors occurring. When run with gdb no errors are found. Valgrind does state that 60 bytes of memory were definitely lost in 1 block showing that we forgot to free those bytes. (gcc -g -o p4 p4.c)
5. When running this normally no errors occur but when running this with valgrind we find that we tried to write four bytes to an invalid memory location and it tells us that at the memory location it found 0 bytes after 400 bytes was allocated to it showing us the invalid write operation we performed. This program is incorrect because it accesses memory outside of the bounds of the allocated block. (gcc -g -o data data.c)
6. The program does run but prints a random integer rather than the one assigned before freeing the array. When run with valgrind it states that there was an invalid read of size 4 in main indicating the reading of a memory address that has been freed. (gcc -g -o p6 p6.c)
7. When I compiled my code that freed a pointer to a random piece of the array allocated by malloc it gave a warning stating “warning: ‘free’ called on pointer ‘arr’ with nonzero offset 12” But I was still able to run which then gave me the output “free(): invalid pointer Aborted (core dumped)”. Therefore you do not need extra tools to find that you free’d the wrong piece of memory by accident. (gcc -g -o p7 p7.c)