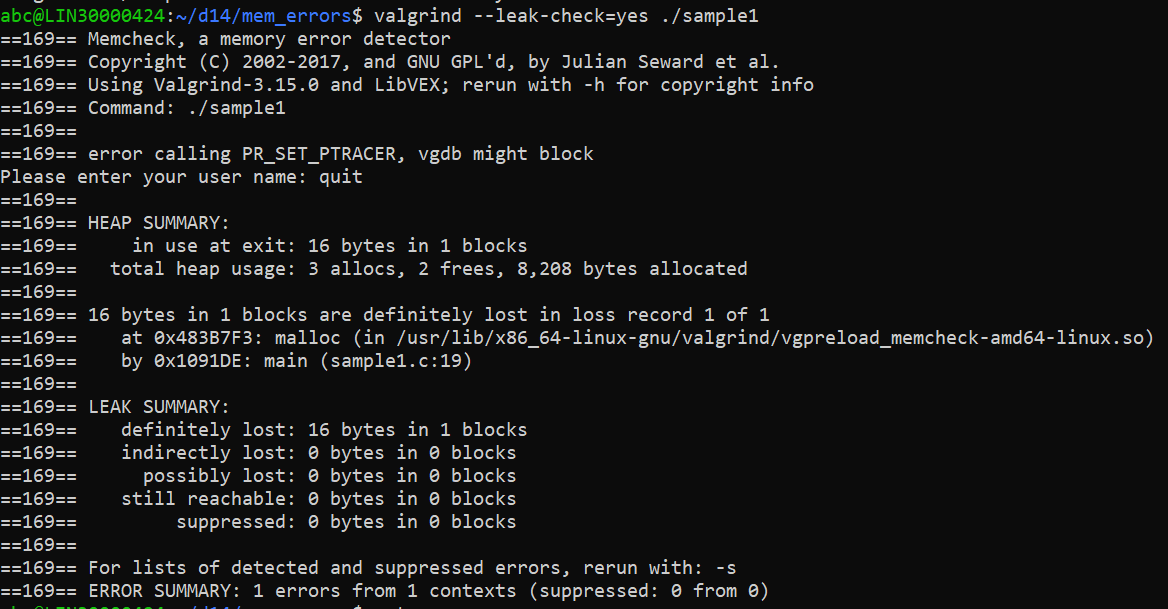
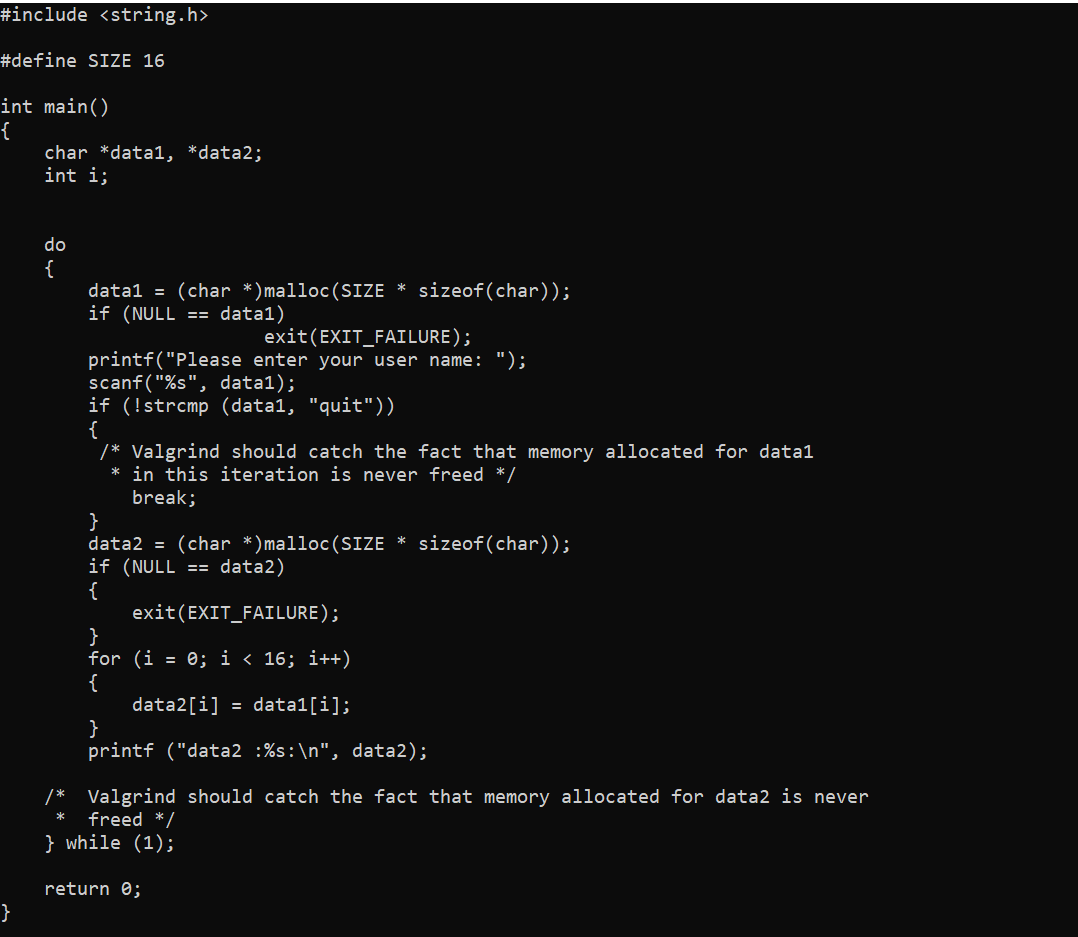
1st Program

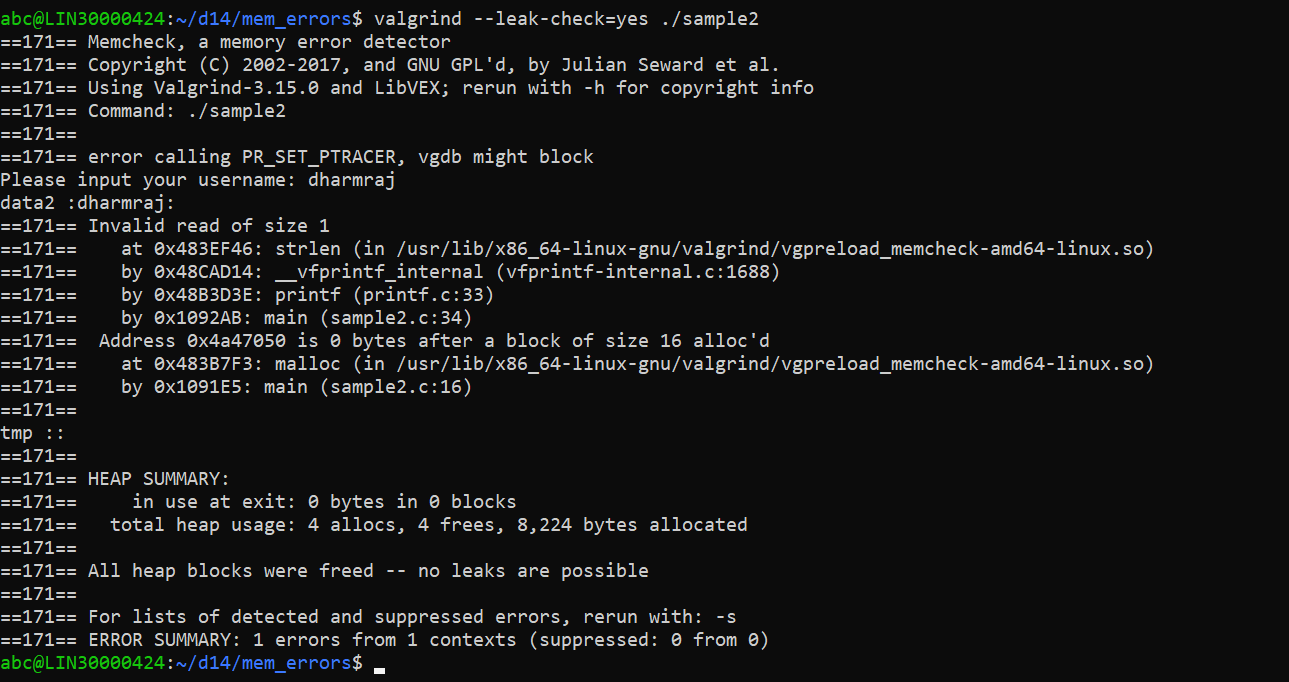




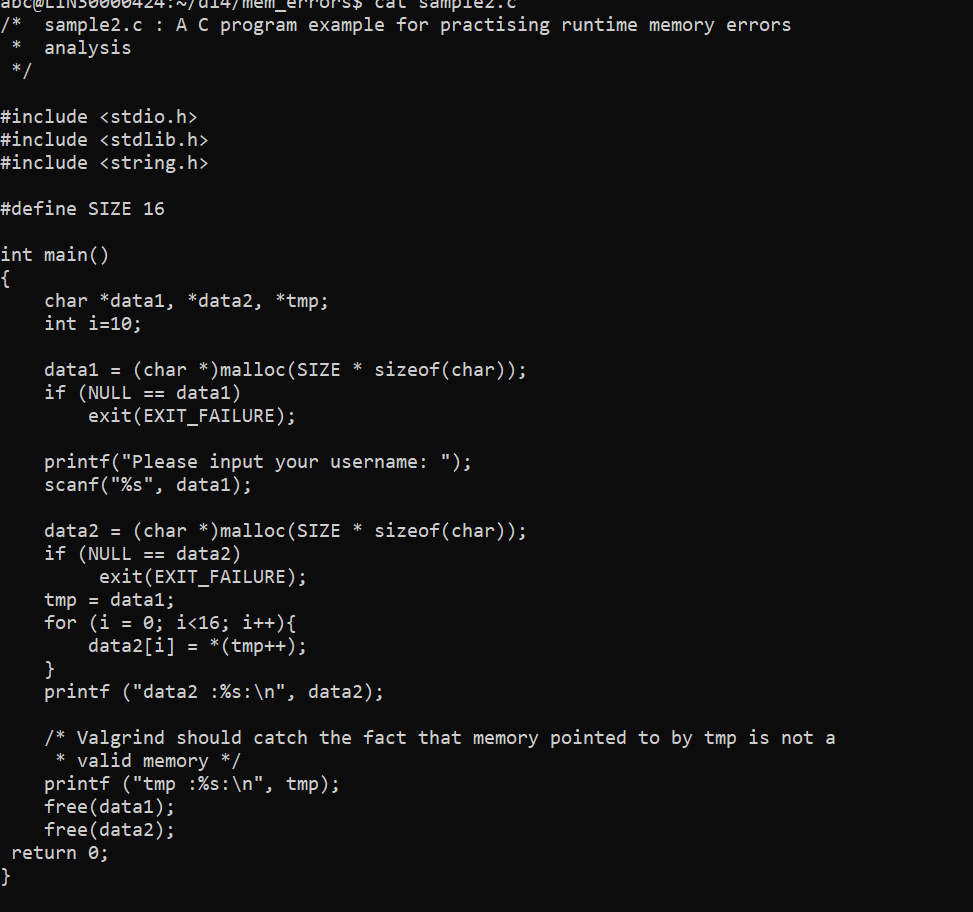
It shows definitely lost coz in the last they didn’t freed the pointer.

As per the definition:- "Definitely lost". This covers case 3 (for the BBB blocks) above. This means that no pointer to the block can be found. The block is classified as "lost", because the programmer could not possibly have freed it at program exit, since no pointer to it exists. This is likely a symptom of having lost the pointer at some earlier point in the program.

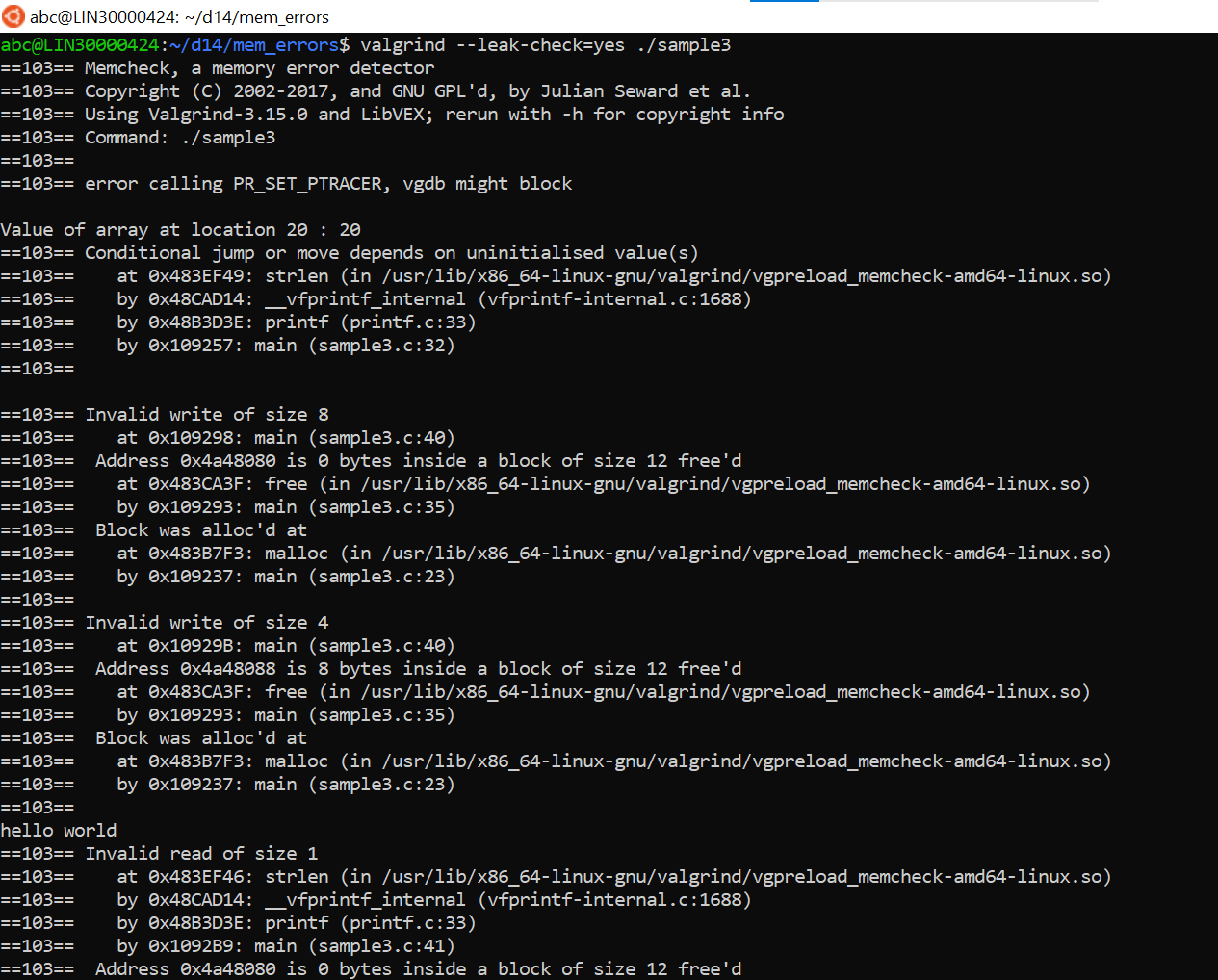
2nd program.

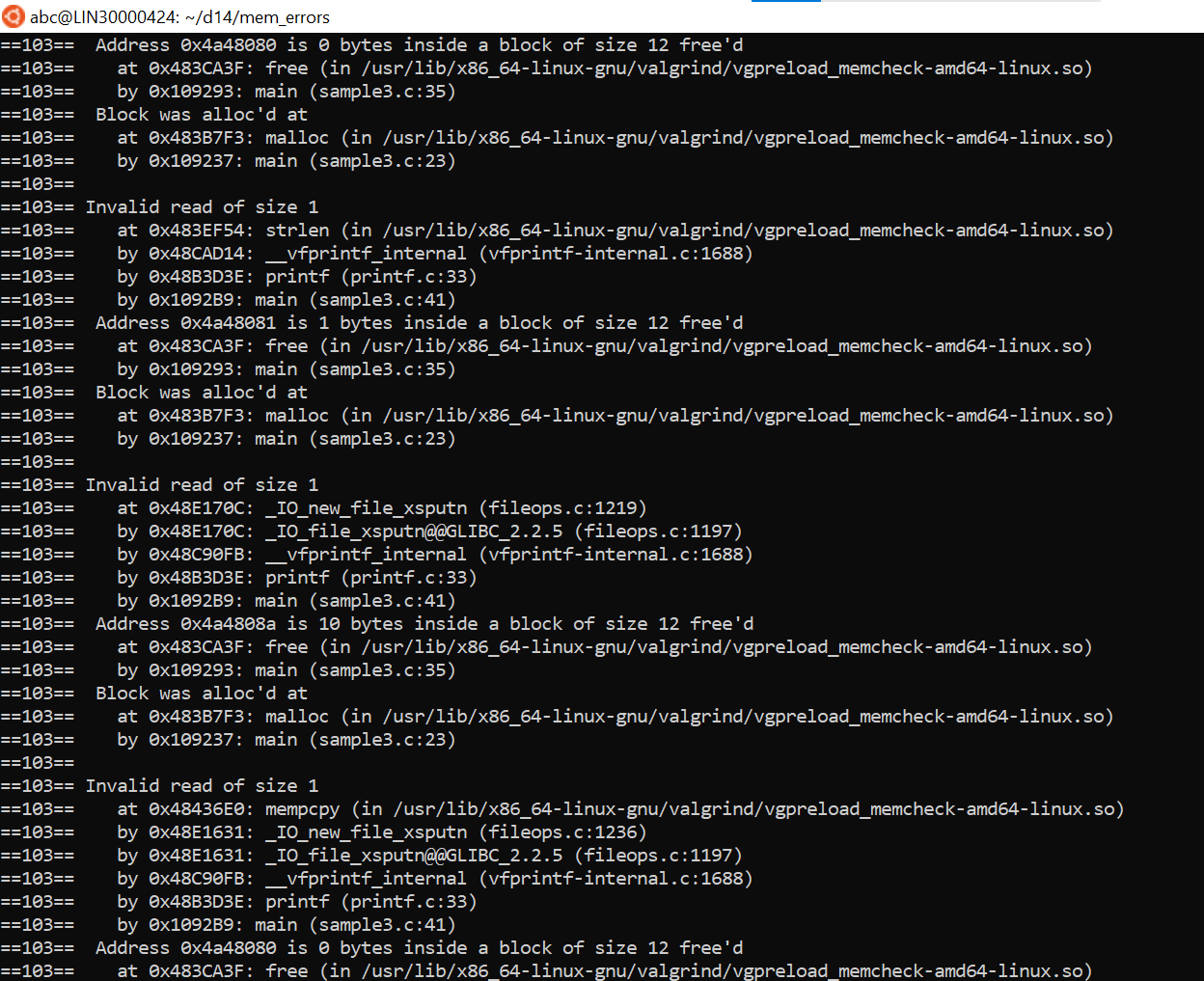
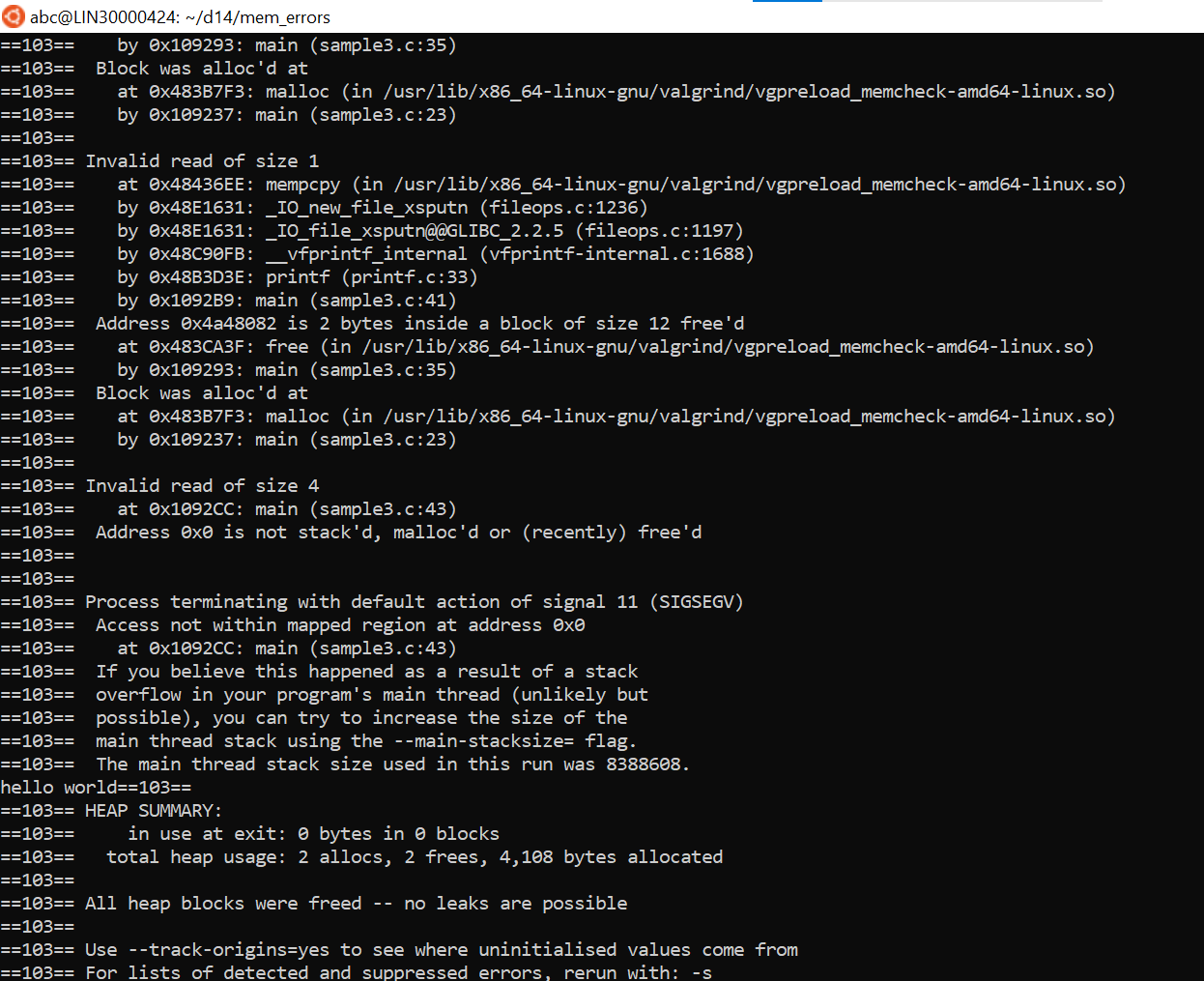


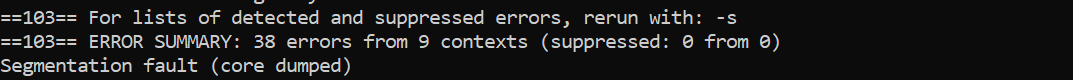
379039 syscall wrapper for prctl(PR\_SET\_NAME) must not check more than 16 bytes



Program 3

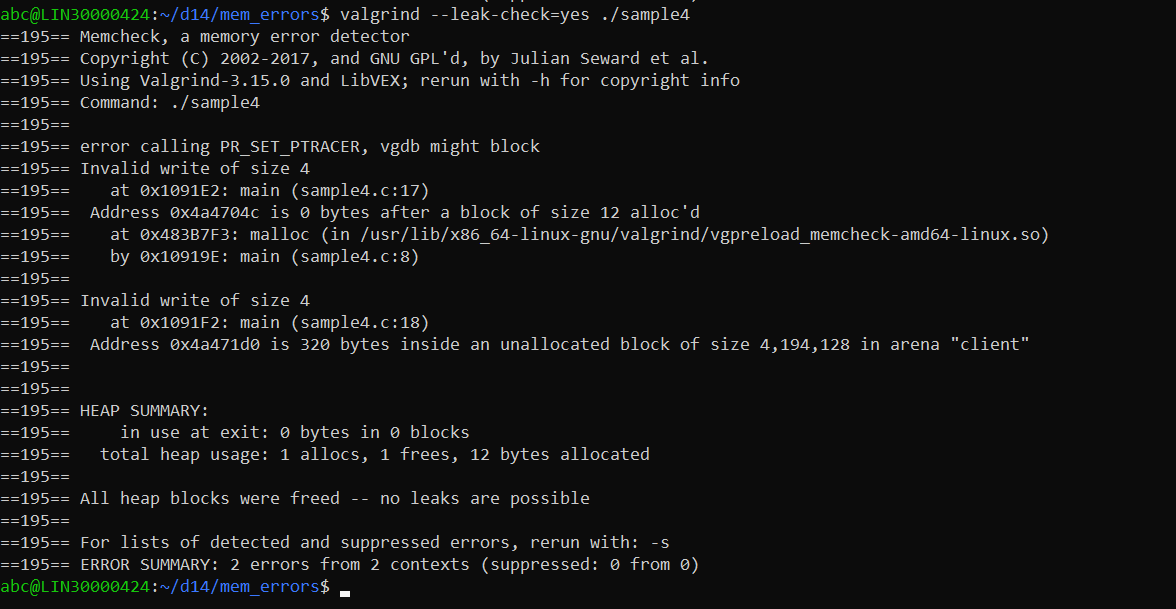
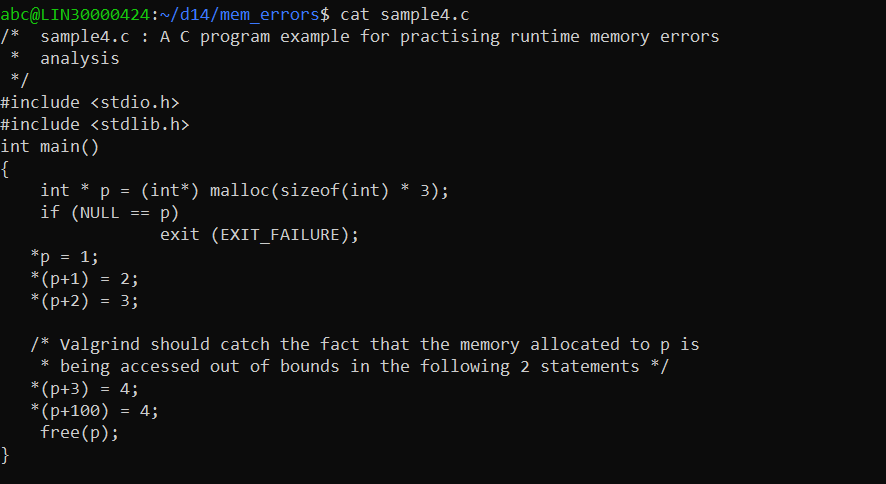






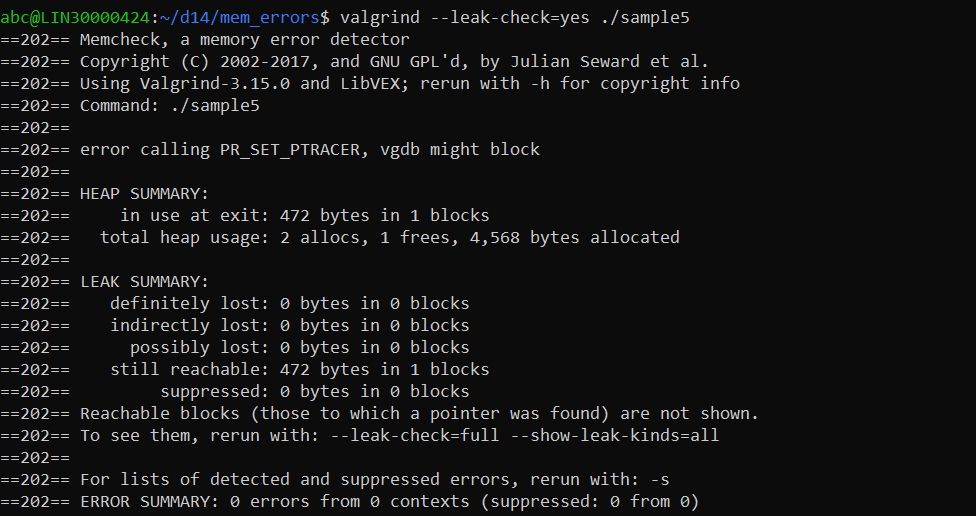
* Valgrind does not catch the fact that arr is being accessed out of bounds This is a limitation of the valgrind tool:- in given code it’s:- arr[i+10] = i+10
* Valgrind should catch the fact that a is accessed without initialization
* Valgrind should catch the fact that contents of p are accessed by printf without being initialized
* Valgrind should catch the fact that contents of p are accessed after p has been freed
* Valgrind should catch the fact that address of a local variable is returned. This is invalid as the scope and lifetime of a local variable which is allocated memory from the stack is only till the function returns

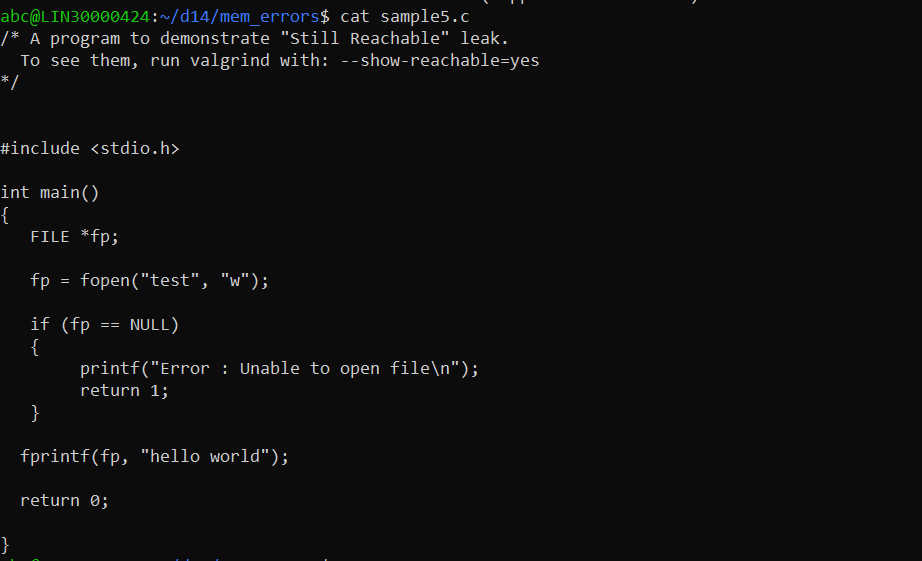
Program 4

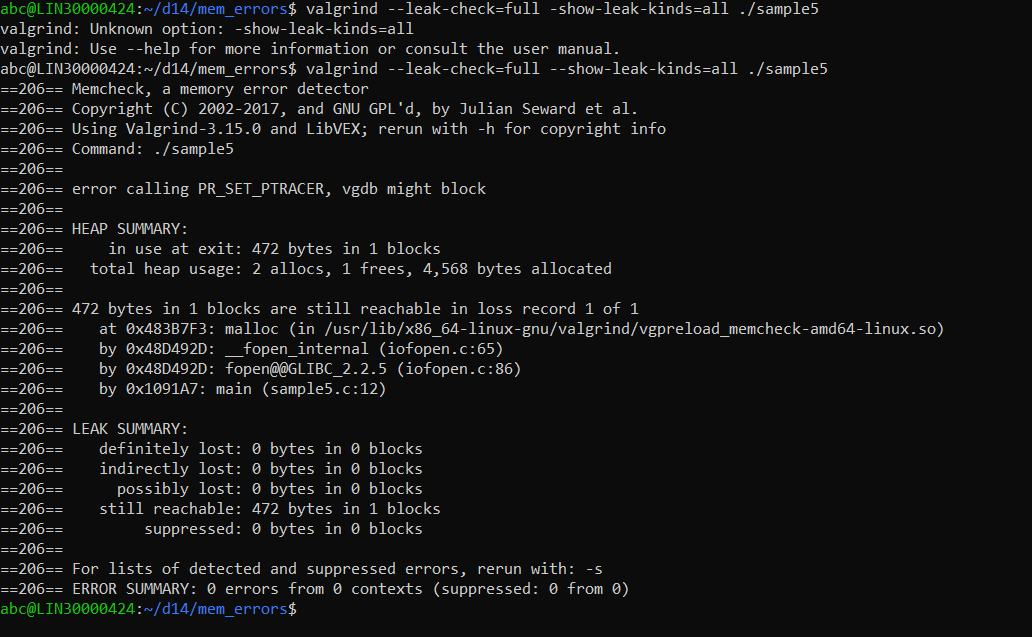


Coz of out of bound 2 errors at write time happen.

Program 5







TYhe Memcheck monitor command leak\_check full reachable any. This requests a full reporting of the allocated memory blocks

GDB will send the leak\_check command to the Valgrind gdbserver. The Valgrind gdbserver will execute the

monitor command itself, if it recognises it to be a Valgrind core monitor command. If it is not recognised as such, it

is assumed to be tool-specific and is handed to the tool for execution.