Valgrind report

Valgrind is an open-source memory access error and leak detection tool. An optimization level of 1 is generally faster than level 0, although it can cause incorrect line numbers to be reported. An optimization level higher than 1 can cause spurious uninitialised-value errors to be reported

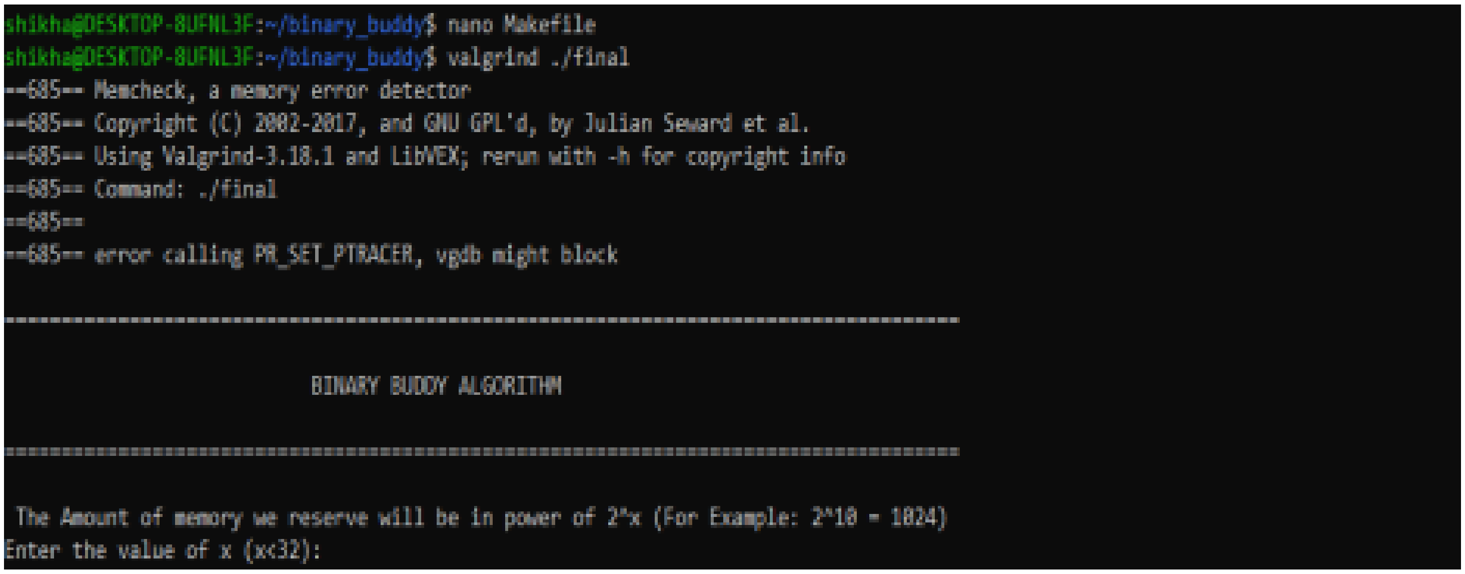
How to compile code for valgrind?

```
$ g++ main.cpp -o final
```

How to run a program under valgrind

```
$ valgrind ./final
```

Sample output report of valgrind



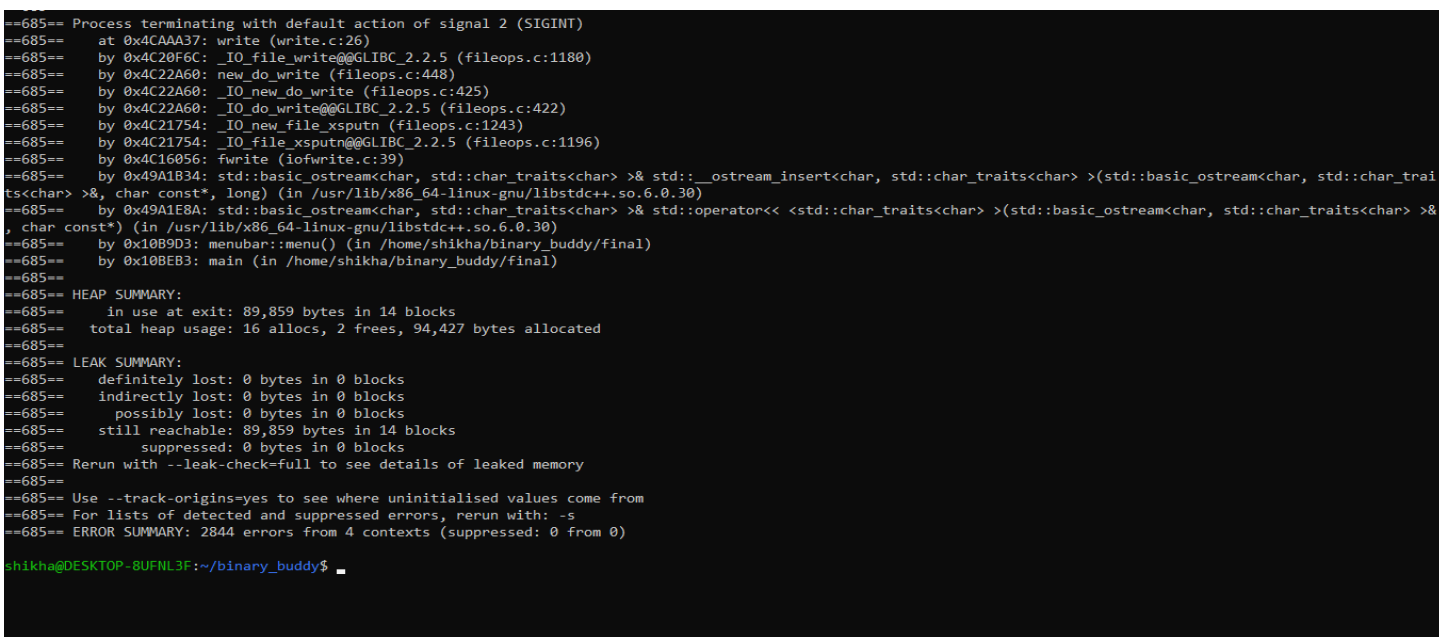
```
shikha@DESKTOP-8UFNL3F:~/binary_buddy$ nano Makefile
shikha@DESKTOP-8UFNL3F:~/binary_buddy$ valgrind ./final
==685== Memcheck, a memory error detector
==685== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==685== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==685== Command: ./final
==685==
==685== error calling PR_SET_PTRACER, vgdb might block

===========================================================================

                        BINARY BUDDY ALGORITHM

===========================================================================

 The Amount of memory we reserve will be in power of 2^x (For Example: 2^10 = 1024)
Enter the value of x (x<32):
```



```
==685== Process terminating with default action of signal 2 (SIGINT)
==685==    at 0x4CAAA37: write (write.c:26)
==685==    by 0x4C20F6C: _IO_file_write@@GLIBC_2.2.5 (fileops.c:1180)
==685==    by 0x4C22A60: new_do_write (fileops.c:448)
==685==    by 0x4C22A60: _IO_new_do_write (fileops.c:425)
==685==    by 0x4C22A60: _IO_do_write@@GLIBC_2.2.5 (fileops.c:422)
==685==    by 0x4C21754: _IO_new_file_xsputn (fileops.c:1243)
==685==    by 0x4C21754: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1196)
==685==    by 0x4C16056: fwrite (iofwrite.c:39)
==685==    by 0x49A1B34: std::basic_ostream<char, std::char_traits<char> >& std::__ostream_insert<char, std::char_traits<char> >(std::basic_ostream<char, std::char_trai
ts<char> >&, char const*, long) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30)
==685==    by 0x49A1E8A: std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&
, char const*) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30)
==685==    by 0x10B9D3: menubar::menu() (in /home/shikha/binary_buddy/final)
==685==    by 0x10BEB3: main (in /home/shikha/binary_buddy/final)
==685==
==685== HEAP SUMMARY:
==685==     in use at exit: 89,859 bytes in 14 blocks
==685==   total heap usage: 16 allocs, 2 frees, 94,427 bytes allocated
==685==
==685== LEAK SUMMARY:
==685==    definitely lost: 0 bytes in 0 blocks
==685==    indirectly lost: 0 bytes in 0 blocks
==685==      possibly lost: 0 bytes in 0 blocks
==685==    still reachable: 89,859 bytes in 14 blocks
==685==         suppressed: 0 bytes in 0 blocks
==685== Rerun with --leak-check=full to see details of leaked memory
==685==
==685== Use --track-origins=yes to see where uninitialised values come from
==685== For lists of detected and suppressed errors, rerun with: -s
==685== ERROR SUMMARY: 2844 errors from 4 contexts (suppressed: 0 from 0)

shikha@DESKTOP-8UFNL3F:~/binary_buddy$
```