VALGRIND REPORT

CSP301: PRODUCT 2

SOCIAL NETWORK SIMULATOR

Created By:

Devansh Dalal
2012CS10224

Poojan Nikunjkumar Mehta
2012CS10241

Shubhankar Suman Singh
2012CS10255

ANALYSIS REPORT

GENERATOR

Our Checking Procedure:

1. Running generator for different number of days
    a) 1000
    b) 1500
    c) 2000
    e) 2500
    f) 3000
    g) 3500
    h) 4000
    i) 5000.

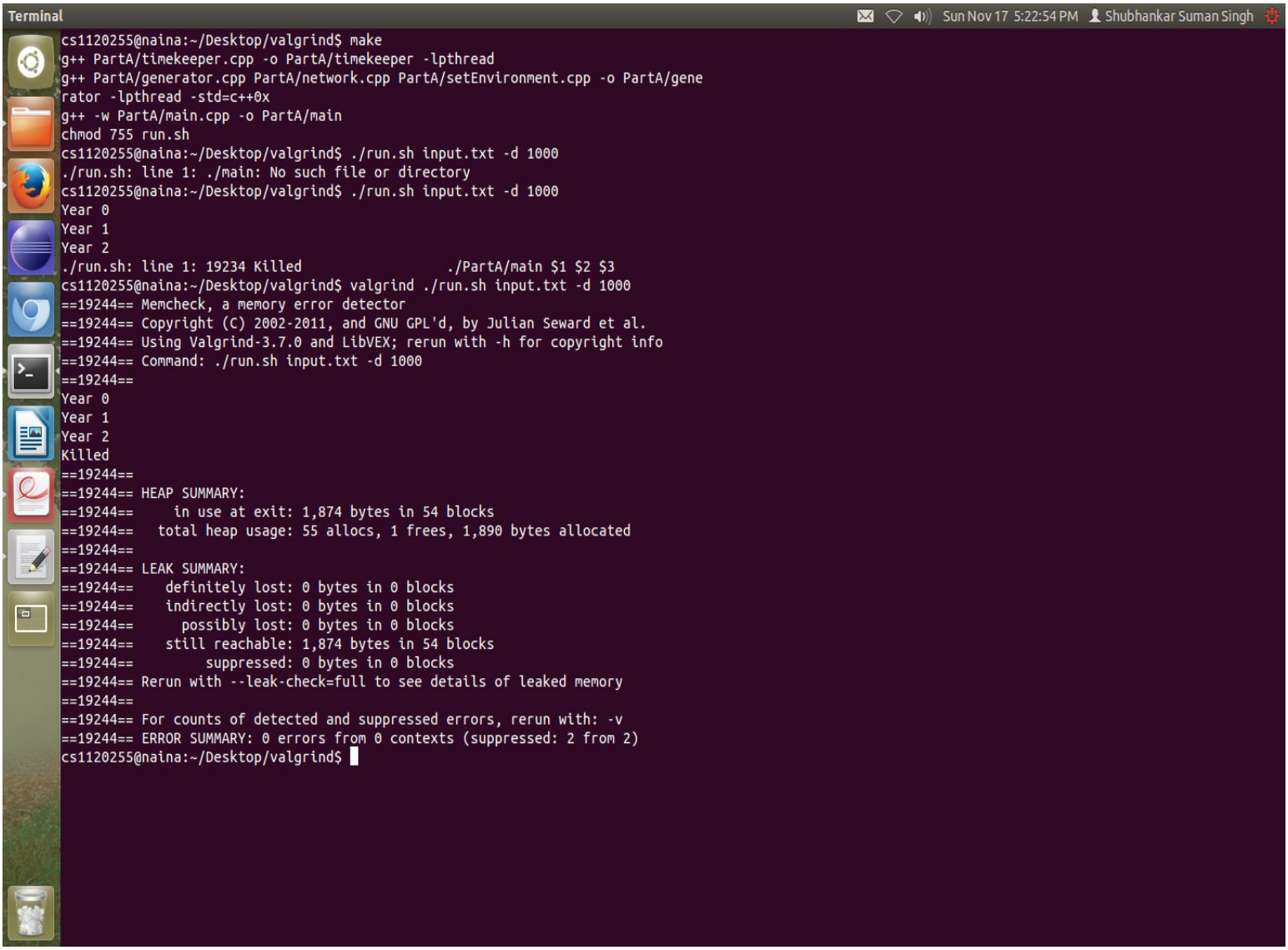
2. Running generator for different number of Nodes (Faculty + Students)
    a) 500
    b) 1000
    c) 2000
    d) 3000
    e) 4000
    f) 5000

We checked for all the above cases and we get same results in all the runs.

- **STILL REACHABLE:**

A block of memory is reported to be Still Reachable when Memcheck finds, after process execution ends, at least one pointer with the start address of the block (a start-pointer). This pointer found can be either in stack (a global pointer, or a pointer in the main function), or in heap if it was referenced by another start-pointer (which again could be in the stack, or referenced by another start-pointer, thus forming a chain of start-pointers). Usually, 'Still Reachable' memory leaks are not considered harmful in the standard development. There are stable and popular libraries in the Free Software world (talking about GLib and GTK+ here) which dynamically allocate a lot of heap memory for data which should be available during the whole execution of the program; this is, memory which shouldn't be deallocated until program is about to exit. And in this situation, instead of explicitly freeing this memory before the return of the main function, it's just easier and faster to leave the kernel do it automatically when the process ends.

Results as seen by using VALGRIND tool:



- **DEFINITELY LOST:**
  A leak is considered Definitely Lost when, at process exit, there is no pointer or chain of pointers to the leaked memory block.

- **HEAP SUMMARY**:

This report shows some statistical values of the execution
Memory in use at exit : 1,874 bytes in 54 blocks
Total heap usage : 55 allocs, 1 frees, 1,874 bytes allocated

- **LEAK SUMMARY**:

  This report shows about memory leakage in the program:
  definitely lost : 0 bytes in 0 blocks
  indirectly lost : 0 bytes in 0 blocks
  possibly lost : 0 bytes in 0 blocks
  still reachable : 1,874 bytes in 54 blocks
  suppressed : 0 bytes in 0 blocks

- **ERROR SUMMARY**:

  0 errors from 0 contexts (suppressed: 2 from 2)

- # GYANI

  Our Checking Procedure:

  (1) We run Gyani for different graphml files generated.
  (2) Then we ask various queries and finally quit.

- **HEAP SUMMARY**:

  This report shows some statistical values of the execution:
  Memory in use at exit : 1,902 bytes in 55 blocks
  Total heap usage : 56 allocs, 1 frees, 1,918 bytes allocated

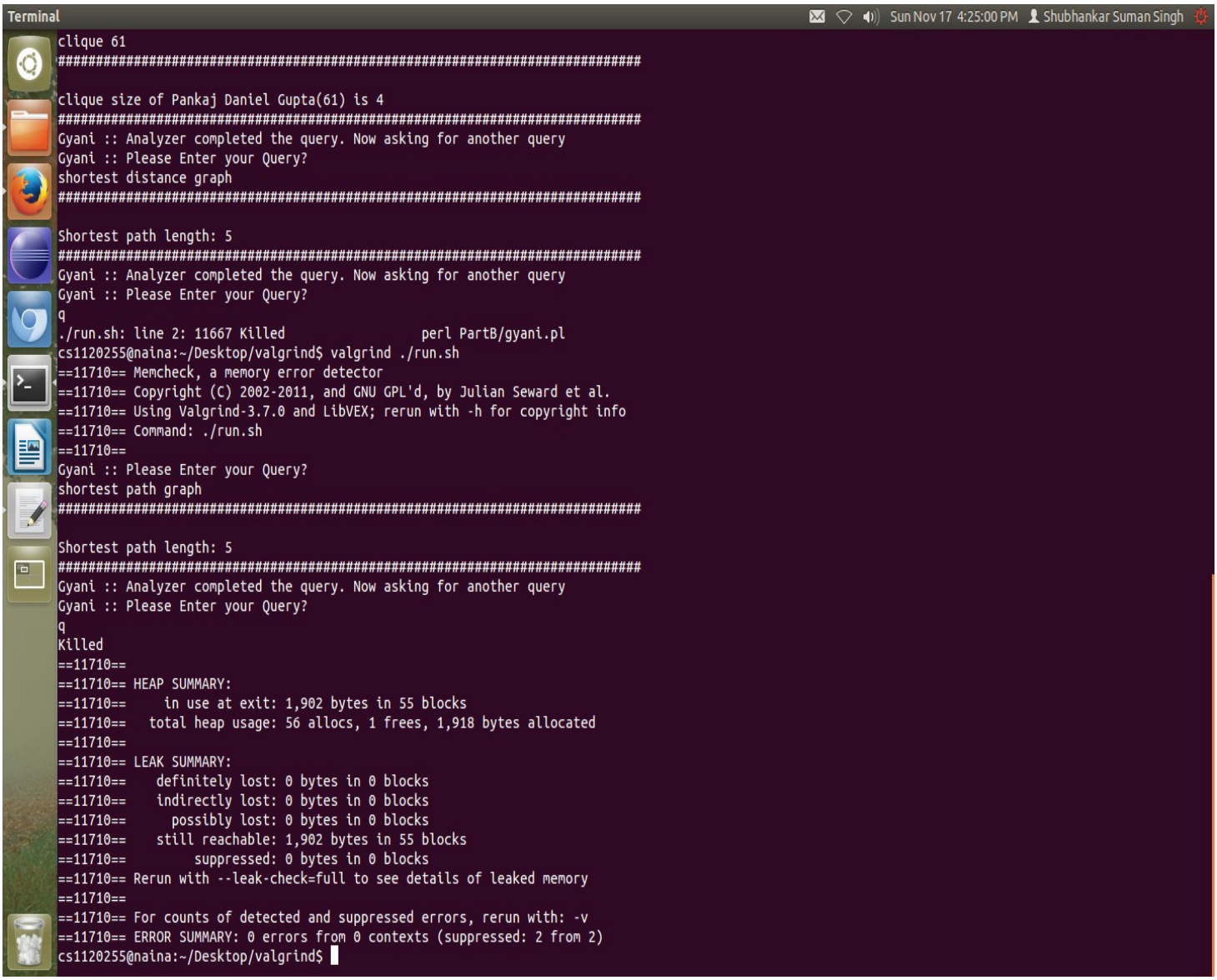
- **LEAK SUMMARY**:

  This report shows about memory leakage in the program
  definitely lost : 0 bytes in 0 blocks
  indirectly lost : 0 bytes in 0 blocks
  possibly lost : 0 bytes in 0 blocks
  still reachable : 1,902bytes in 55 blocks
  suppressed : 0 bytes in 0 blocks

- **ERROR SUMMARY**:

0 errors from 0 contexts (suppressed: 2 from 2)

Results: The results are always same in all the cases.



CONCLUSION

1) There are no memory leaks as the definitely lost, indirectly lost, possibly lost memory values are 0.
2) Also, there are no errors being shown.
3) Only, the valgrind shows still reachable memory, which is due to data being used during the execution and they should not be removed till the exit. Also, the library function uses memory in the heap. The Kernel automatically frees the memory once the program ends.
4) Thus, we have a good memory usage with **zero** memory leaks.