1.

a.

[UC syed.kirmani1@csx3 stutter] time python stutter.py < t3.txt
Longest stutter: ___o.O.o_____o.O.o___

real   0m0.037s
user   0m0.023s
sys   0m0.011s

[UC syed.kirmani1@csx3 stutter] time slow-stut < t3.txt
Longest stutter: ___o.O.o_____o.O.o___

real   0m0.011s
user   0m0.005s
sys   0m0.004s

[UC syed.kirmani1@csx3 stutter] time python stutter.py < t4.txt
Longest stutter: Tartar

real   0m0.260s
user   0m0.245s
sys   0m0.010s

[UC syed.kirmani1@csx3 stutter] time slow-stut < t4.txt
Longest stutter: Tartar

real   0m2.357s
user   0m0.369s
sys   0m1.973s

b. The Python program for txt3 took 0.023s, for txt4 it took 0.245s on the CPU. For txt3 it took 0.014s and for txt4 it took 0.015s for I/O.
The C++ program for txt3 took 0.005s, for txt4 it took 0.369s on the CPU. For txt3 it took 0.06s and for txt4 it took 1.988s for I/O.

The I/O time is calculated assuming that on idle all system calls are I/O calls.

c.

[UC syed.kirmani1@csx3 stutter] strace -c python stutter.py < t3.txt
Longest stutter: ___o.O.o_____o.O.o___

% time	seconds	usecs/call	calls	errors	syscall
32.51	0.001481	5	290	108	newfstatat
16.70	0.000761	11	65		read
14.16	0.000645	645	1		execve

10.32	0.000470	9	51	7	openat
10.16	0.000463	17	26		mmap
2.88	0.000131	2	44		close
2.19	0.000100	20	5		mprotect
1.89	0.000086	4	18		getdents64
1.58	0.000072	1	55	2	lseek
1.58	0.000072	1	66		rt_sigaction
1.25	0.000057	1	34	29	ioctl
0.88	0.000040	10	4		munmap
0.83	0.000038	3	12		brk
0.44	0.000020	4	5	3	readlink
0.40	0.000018	9	2		pread64
0.33	0.000015	15	1		write
0.29	0.000013	6	2	1	arch_prctl
0.26	0.000012	6	2		getcwd
0.26	0.000012	12	1		set_tid_address
0.26	0.000012	12	1		rseq
0.24	0.000011	11	1		set_robust_list
0.20	0.000009	4	2		getrandom
0.13	0.000006	1	4		fcntl
0.11	0.000005	5	1		prlimit64
0.09	0.000004	4	1		futex
0.07	0.000003	3	1		gettid
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		sysinfo
0.00	0.000000	0	1		getuid
0.00	0.000000	0	1		getgid
0.00	0.000000	0	1		geteuid
0.00	0.000000	0	1		getegid
100.00	0.004556	6	701	151	total

[UC syed.kirmani1@csx3 stutter] strace -c slow-stut < t3.txt
Longest stutter: ___o.O.o_____o.O.o___

% time	seconds	usecs/call	calls	errors	syscall
71.68	0.000858	11	76		read
7.94	0.000095	15	6		mprotect
5.18	0.000062	2	23		mmap
2.84	0.000034	34	1		munmap
2.76	0.000033	11	3		brk
1.34	0.000016	16	1		write
1.09	0.000013	13	1		getrandom
1.00	0.000012	12	1		futex

```
  1.00    0.000012          12         1           prlimit64
  1.00    0.000012          12         1           rseq
  0.92    0.000011           2         5           close
  0.92    0.000011           5         2         1 arch_prctl
  0.92    0.000011          11         1           set_tid_address
  0.84    0.000010           1         6           newfstatat
  0.58    0.000007           7         1           set_robust_list
  0.00    0.000000           0         2           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         5           openat
------ ----------- ----------- --------- --------- ----------------
100.00    0.001197           8       138         2 total

[UC syed.kirmani1@csx3 stutter] strace -c python stutter.py < t4.t
xt
Longest stutter: Tartar
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 31.82    0.001014          11        87           read
 16.91    0.000539          20        26           mmap
 15.25    0.000486           1       290       108 newfstatat
  8.60    0.000274           5        51         7 openat
  6.65    0.000212          11        18           getdents64
  4.61    0.000147           3        44           close
  3.51    0.000112          22         5           mprotect
  2.32    0.000074           6        12           brk
  1.88    0.000060           1        55         2 lseek
  1.41    0.000045          11         4           munmap
  1.07    0.000034           1        34        29 ioctl
  0.88    0.000028          14         2           pread64
  0.88    0.000028          14         2           getrandom
  0.60    0.000019           9         2           getcwd
  0.60    0.000019           3         5         3 readlink
  0.41    0.000013          13         1           prlimit64
  0.38    0.000012           6         2         1 arch_prctl
  0.38    0.000012          12         1           gettid
  0.38    0.000012          12         1           futex
  0.38    0.000012          12         1           set_tid_address
  0.35    0.000011          11         1           set_robust_list
  0.35    0.000011          11         1           rseq
  0.25    0.000008           8         1           sysinfo
  0.16    0.000005           1         4           fcntl
  0.00    0.000000           0         1           write
```

```
  0.00    0.000000           0        66           rt_sigaction
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           getuid
  0.00    0.000000           0         1           getgid
  0.00    0.000000           0         1           geteuid
  0.00    0.000000           0         1           getegid
------ ----------- ----------- --------- --------- ----------------
100.00    0.003187           4       723       151 total
```

```
[UC syed.kirmani1@csx3 stutter] strace -c slow-stut < t4.txt
Longest stutter: Tartar
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00   26.450427           4   5767198           read
  0.00    0.000280          12        23           mmap
  0.00    0.000072          12         6           mprotect
  0.00    0.000033           6         5           openat
  0.00    0.000028           4         6           newfstatat
  0.00    0.000022           4         5           close
  0.00    0.000016          16         1           munmap
  0.00    0.000013           4         3           brk
  0.00    0.000011          11         1           write
  0.00    0.000011           5         2           pread64
  0.00    0.000007           7         1           set_tid_address
  0.00    0.000005           5         1           set_robust_list
  0.00    0.000004           2         2         1 arch_prctl
  0.00    0.000004           4         1           futex
  0.00    0.000004           4         1           prlimit64
  0.00    0.000004           4         1           getrandom
  0.00    0.000003           3         1           rseq
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
------ ----------- ----------- --------- --------- ----------------
100.00   26.450944           4   5767260         2 total
```

d. The stutter.py Python code has a buffer already implemented. This allows the code to process large chunks of data all at once and so executions on large inputs come out to be faster than the slow-stut.cpp file. The buffer is reducing the number of read system calls since it is not just reading one character at a time like the C++ code. This can be seen with the execution of t3 and t4 in the above table showing that the C++ code has much higher read system calls. The Python file could also be slower on some smaller inputs since it's an interpreted

language which introduces some overheard. While C++ is a compiled language so everything is compiled before execution.

2. Code
3.
    a. [UC syed.kirmani1@csx1 stutter] time fast-stut < t3.txt
    Longest stutter: ___o.O.o_____o.O.o___

    real 0m0.010s
    user 0m0.004s
    sys 0m0.005s

    [UC syed.kirmani1@csx1 stutter] time fast-stut < t4.txt
    Longest stutter: Tartar

    real 0m0.125s
    user 0m0.113s
    sys 0m0.010s

    [UC syed.kirmani1@csx1 stutter] strace -c fast-stut < t3.txt
    Longest stutter: ___o.O.o_____o.O.o___

% time	seconds	usecs/call	calls	errors	syscall
58.01	0.000565	24	23		mmap
8.01	0.000078	13	6		mprotect
7.60	0.000074	12	6		read
7.19	0.000070	14	5		openat
6.16	0.000060	10	6		newfstatat
5.13	0.000050	10	5		close
2.87	0.000028	14	2		pread64
1.33	0.000013	6	2	1	arch_prctl
1.33	0.000013	13	1		set_tid_address
1.23	0.000012	12	1		set_robust_list
1.13	0.000011	11	1		rseq
0.00	0.000000	0	1		write
0.00	0.000000	0	1		munmap
0.00	0.000000	0	3		brk
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		execve
0.00	0.000000	0	1		futex
0.00	0.000000	0	1		prlimit64
0.00	0.000000	0	1		getrandom
100.00	0.000974	14	68	2	total

```
[UC syed.kirmani1@csx1 stutter]  strace -c fast-stut < t4.txt
Longest stutter: Tartar
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 59.22    0.001480         134        11           read
 24.21    0.000605          26        23           mmap
  3.28    0.000082          13         6           mprotect
  2.92    0.000073          14         5           openat
  1.92    0.000048           9         5           close
  1.80    0.000045           7         6           newfstatat
  1.76    0.000044          44         1           munmap
  1.28    0.000032          10         3           brk
  1.08    0.000027          13         2           pread64
  0.56    0.000014          14         1           getrandom
  0.52    0.000013          13         1           prlimit64
  0.48    0.000012          12         1           futex
  0.24    0.000006           3         2         1 arch_prctl
  0.24    0.000006           6         1           set_tid_address
  0.24    0.000006           6         1           set_robust_list
  0.24    0.000006           6         1           rseq
  0.00    0.000000           0         1           write
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
------ ----------- ----------- --------- --------- ----------------
100.00    0.002499          34        73         2 total
```

b. Yes, my fast-stut.cpp is faster than slow-stut.cpp. I believe this is due to the drastic decrease in read system calls upon addition of a larger buffer that processes more than one character at a time. From the table in 1c for t4.txt on slow-stut.cpp the read system calls went from 5767198 to 11 in fast-stut.cpp. This overall improved the run time drastically.
c. Yes my fast-stut.cpp is faster than the Python stutter.py file. This can be shown upon inspection of the time command on both t3 and t4 in question 1a. The runtime for my fast-stut on t3 and t4 is much faster than the runtime of t3 and t4 with stutter.py (0.010s vs 0.037s and 0.125s vs 0.260s).