Question 1

a) **Python (palindrome.py)**

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ time python3 palindrome.py < t3.txt
Longest palindrome: ___o.O.o___

real    0m0.018s
user    0m0.016s
sys     0m0.003s
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ time python3 palindrome.py < t4.txt
Longest palindrome: redder

real    0m0.251s
user    0m0.243s
sys     0m0.006s
```

C++ (slow-pali.cpp)

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ time ./slow-pali < t3.txt
Longest palindrome: ___o.O.o___

real    0m0.008s
user    0m0.002s
sys     0m0.003s
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ time ./slow-pali < t4.txt
Longest palindrome: redder

real    0m2.470s
user    0m1.181s
sys     0m1.287s
```

b) Referring back to Q1 A), user is time the program spent in user mode while sys is time the program spent in kernel mode

c) Python is a higher-level language where the system has to interpret the code before it can be compiled compared to C++ which is a lower-level language and can directly be compiled so is inherently faster. In addition to this, if we refer to the strace outputs for t3.txt in Q3 A) and B), we can see that for the C++ implementation (slow-pali.cpp) there are 50 read calls compared to the python implementation's (palindrome.py) 84, so it is fairly clear cut in this instance why the C++ implementation was faster in addition to having an advantage in being more efficient when it comes to system calls. Now in the case of the strace outputs for t4.txt, we see the C++ implementation making over 5 million read calls vs just 788 from the python implementation, for which the time outputs from Q1 A) also just confirm that the python implementation is faster (due to being more optimized) on a larger input, although python as a language is slower. All in all, the python implementation is more efficient with larger input sizes compared to the C++ implementation being faster for smaller input sizes

Question 2

Refer to *fast-pali.cpp*

Question 3

a) Based on the time results from comparing slow-pali.cpp to fast-pali.cpp (seen below), we know that the latter is faster than the former when it comes to larger-sized inputs. To support this result, we can also refer to the t3.txt strace results from the two where we see slow-pali.cpp making 50 read calls compared to 14 for fast-pali.cpp and the results being similar for t4.txt as well.

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ time ./fast-pali < t3.txt
Longest palindrome: ___o.O.o___

real    0m0.009s
user    0m0.002s
sys     0m0.004s
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ time ./fast-pali < t4.txt
Longest palindrome: redder

real    0m0.076s
user    0m0.071s
sys     0m0.005s
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ strace -c ./slow-pali < t3.txt
Longest palindrome: ___o.O.o___
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 38.18    0.000113           2        48        43 openat
 21.28    0.000063           2        22           mmap
 18.92    0.000056           1        50           read
  5.74    0.000017           2         8         7 stat
  4.39    0.000013           1         7           lseek
  3.72    0.000011           1         6           fstat
  3.38    0.000010           1         7           mprotect
  2.03    0.000006           1         5           close
  1.01    0.000003           3         1           munmap
  0.68    0.000002           2         1           write
  0.68    0.000002           0         3           brk
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.000296           1       162        52 total
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ strace -c ./slow-pali < t4.txt
Longest palindrome: redder
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    9.639174           1   5767205           read
  0.00    0.000121           2        48        43 openat
  0.00    0.000075           3        22           mmap
  0.00    0.000024           3         7           mprotect
  0.00    0.000018           2         8         7 stat
  0.00    0.000011           1         6           fstat
  0.00    0.000011           1         7           lseek
  0.00    0.000010           3         3           brk
  0.00    0.000010          10         1         1 access
  0.00    0.000009           1         5           close
  0.00    0.000009           4         2         1 arch_prctl
  0.00    0.000008           8         1           munmap
  0.00    0.000006           6         1           write
  0.00    0.000000           0         1           execve
------ ----------- ----------- --------- --------- ----------------
100.00    9.639486           1   5767317        52 total
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ strace -c ./fast-pali < t3.txt
Longest palindrome: ___o.0.o___
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0        14           read
  0.00    0.000000           0         1           write
  0.00    0.000000           0         5           close
  0.00    0.000000           0         8         7 stat
  0.00    0.000000           0         6           fstat
  0.00    0.000000           0         7           lseek
  0.00    0.000000           0        22           mmap
  0.00    0.000000           0         7           mprotect
  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         2         1 arch_prctl
  0.00    0.000000           0        48        43 openat
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000           0       126        52 total
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ strace -c ./fast-pali < t4.txt
Longest palindrome: redder
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 57.55    0.000408          21        19           read
 17.49    0.000124           2        48        43 openat
 11.00    0.000078           3        22           mmap
  3.53    0.000025           3         7           mprotect
  2.68    0.000019           2         8         7 stat
  1.69    0.000012           1         7           lseek
  1.41    0.000010           1         6           fstat
  1.27    0.000009           1         5           close
  0.99    0.000007           7         1           munmap
  0.85    0.000006           2         3           brk
  0.56    0.000004           4         1         1 access
  0.56    0.000004           2         2         1 arch_prctl
  0.42    0.000003           3         1           execve
  0.00    0.000000           0         1           write
------ ----------- ----------- --------- --------- ----------------
100.00    0.000709           5       131        52 total
```

b) Simply by referring to the time outputs from the python implementation (seen below) vs the C++ implementation (fast-pali.cpp as seen in Q3 A)) we can see that the latter is faster. Similarly, if we refer to the strace results for both the programs we can see that the C++ implementation is more efficient based on the number of total read calls being lower.

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ strace -c python3 palindrome.py < t3.txt
Longest palindrome: ___o.O.o___
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 32.00    0.000008           0        18           brk
 28.00    0.000007           0       100           fstat
 16.00    0.000004           0        59           mmap
 12.00    0.000003           0       175        47 stat
  8.00    0.000002           2         1           sysinfo
  4.00    0.000001           0         3           sigaltstack
  0.00    0.000000           0        84           read
  0.00    0.000000           0         1           write
  0.00    0.000000           0        68           close
  0.00    0.000000           0         1           lstat
  0.00    0.000000           0        42         2 lseek
  0.00    0.000000           0        11           mprotect
  0.00    0.000000           0         3           munmap
  0.00    0.000000           0        68           rt_sigaction
  0.00    0.000000           0         1           rt_sigprocmask
  0.00    0.000000           0        18        11 ioctl
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         3           dup
  0.00    0.000000           0         1           getpid
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         3           fcntl
  0.00    0.000000           0         1           getcwd
  0.00    0.000000           0         3         2 readlink
  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
  0.00    0.000000           0         2         1 arch_prctl
  0.00    0.000000           0         2           futex
  0.00    0.000000           0        16           getdents64
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0       141        76 openat
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           prlimit64
  0.00    0.000000           0         1           getrandom
------ ----------- ----------- --------- --------- ----------------
100.00    0.000025           0       835       140 total
```

```
[sharjeel.junaid@gfx-ta3 Assignment1]$ strace -c python3 palindrome.py < t4.txt
Longest palindrome: redder
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 21.91    0.000158           0       788           read
 20.80    0.000150           0       175        47 stat
 16.09    0.000116           1        68           rt_sigaction
  9.43    0.000068           0       100           fstat
  9.43    0.000068           0       141        76 openat
  5.41    0.000039           0        68           close
  4.72    0.000034           0        42         2 lseek
  2.36    0.000017           0        18        11 ioctl
  2.36    0.000017           1        16           getdents64
  2.08    0.000015          15         1           lstat
  1.94    0.000014           0        58           mmap
  1.11    0.000008           8         1           write
  0.97    0.000007           0        62           brk
  0.69    0.000005           1         3           dup
  0.42    0.000003           1         3           fcntl
  0.14    0.000001           1         1           getcwd
  0.14    0.000001           0         3         2 readlink
  0.00    0.000000           0        11           mprotect
  0.00    0.000000           0         2           munmap
  0.00    0.000000           0         1           rt_sigprocmask
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           getpid
  0.00    0.000000           0         1           execve
  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
  0.00    0.000000           0         3           sigaltstack
  0.00    0.000000           0         2         1 arch_prctl
  0.00    0.000000           0         2           futex
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           prlimit64
  0.00    0.000000           0         1           getrandom
------ ----------- ----------- --------- --------- ----------------
100.00    0.000721           0      1581       140 total
```