CPSC 457 Assignment 1 **Donald Huang 30068480**

1.a) palindrome.py execution times:

```
$ time python3 palindrome.py < t3.txt
Longest palindrome: ___o.O.o___

real    0m0.023s
user    0m0.013s
sys     0m0.008s
$ time python3 palindrome.py < t4.txt
Longest palindrome: redder

real    0m0.315s
user    0m0.304s
sys     0m0.009s
```

1.a) slow-pali.cpp execution times:

```
$ time ./slow-pali < t3.txt
Longest palindrome: ___o.O.o___

real    0m0.006s
user    0m0.001s
sys     0m0.004s

$ time ./slow-pali < t4.txt
Longest palindrome: redder

real    0m3.130s
user    0m1.515s
sys     0m1.611s
```

1.b)

palindrome.py – t3.txt
user mode: 0m0.013s
kernel mode: 0m0.008s
kernel mode was 0.005s faster.

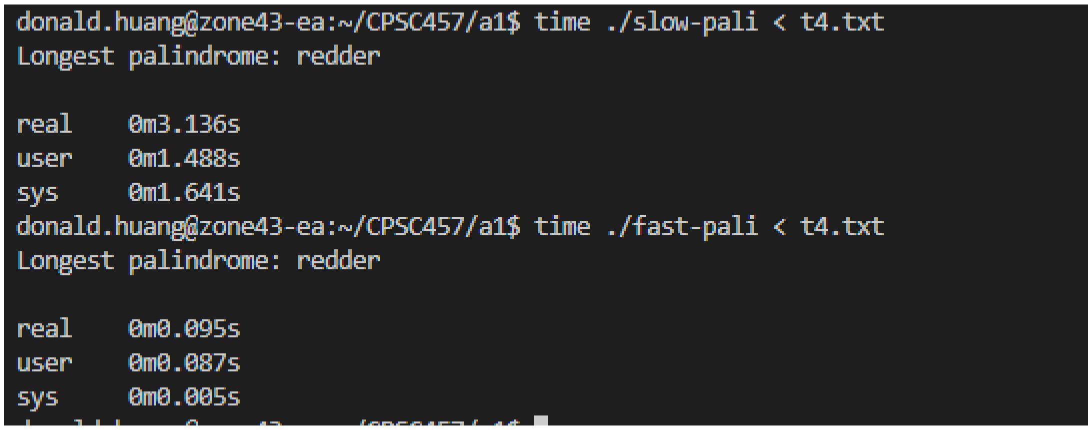
palindrome.py – t4.txt
user mode: 0m0.304s
kernel mode: 0m0.009s
kernel mode was 0.295s faster.

slow-pali.cpp – t3.txt
user mode: 0m0.001s
kernel mode: 0m0.004s
user mode was 0.002s faster.

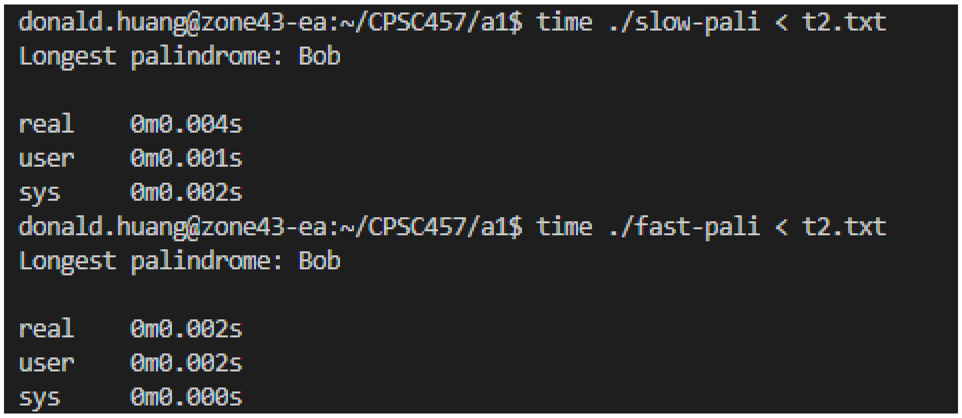
slow-pali.cpp – t4.txt
user mode: 0m1.515s
kernel mode: 0m1.611s
user mode was 0.096s faster.

1.c) If there are lots of short lines with no characters, the c++ will be faster because c++ is a faster language but in the case where python is faster is when there are lines with lots of characters because the words are buffered, and the entire line is read at once whereas the c++ would make lots of calls to the kernel. Overall, it depends on the buffer size that you call and the amount of sys calls. Python makes 1 call to kernel per line, whereas slow-pali makes a kernel call every character.

3.a) Yes my fast-pali.cpp is faster than slow-pali.cpp. This is because in fast we are reading the size of the buffer, and it makes fewer system calls. Here are the results:



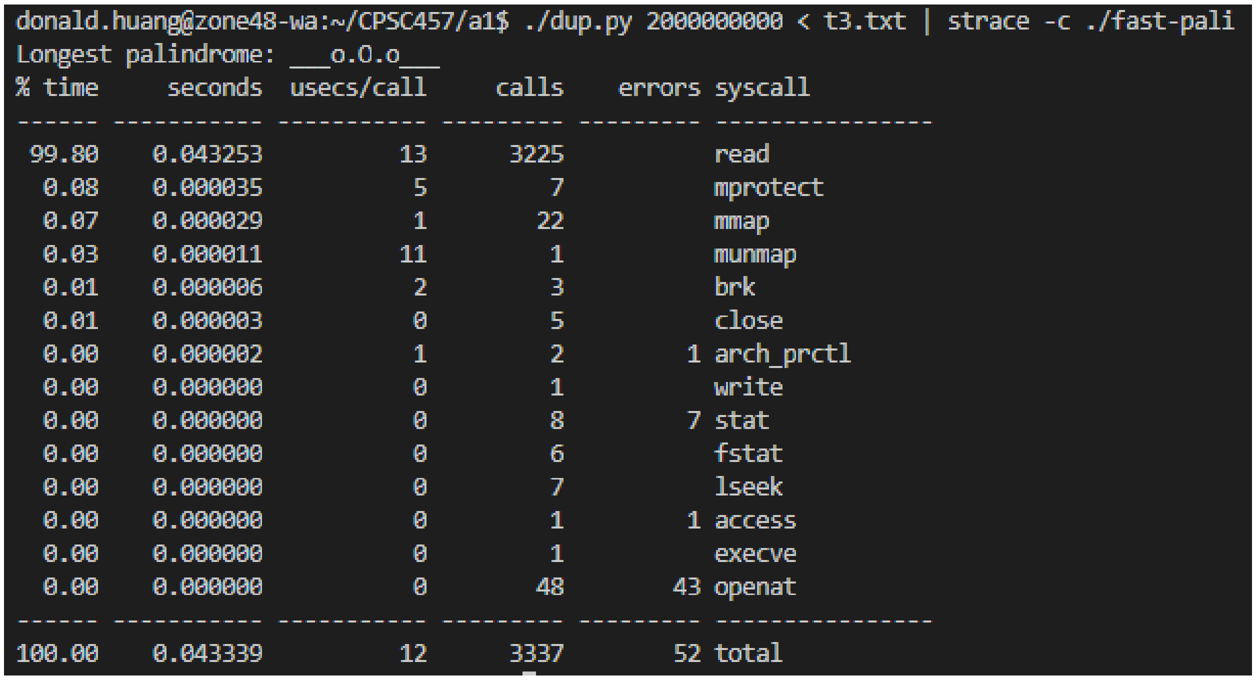
```
donald.huang@zone43-ea:~/CPSC457/a1$ time ./slow-pali < t4.txt
Longest palindrome: redder

real    0m3.136s
user    0m1.488s
sys     0m1.641s
donald.huang@zone43-ea:~/CPSC457/a1$ time ./fast-pali < t4.txt
Longest palindrome: redder

real    0m0.095s
user    0m0.087s
sys     0m0.005s
```



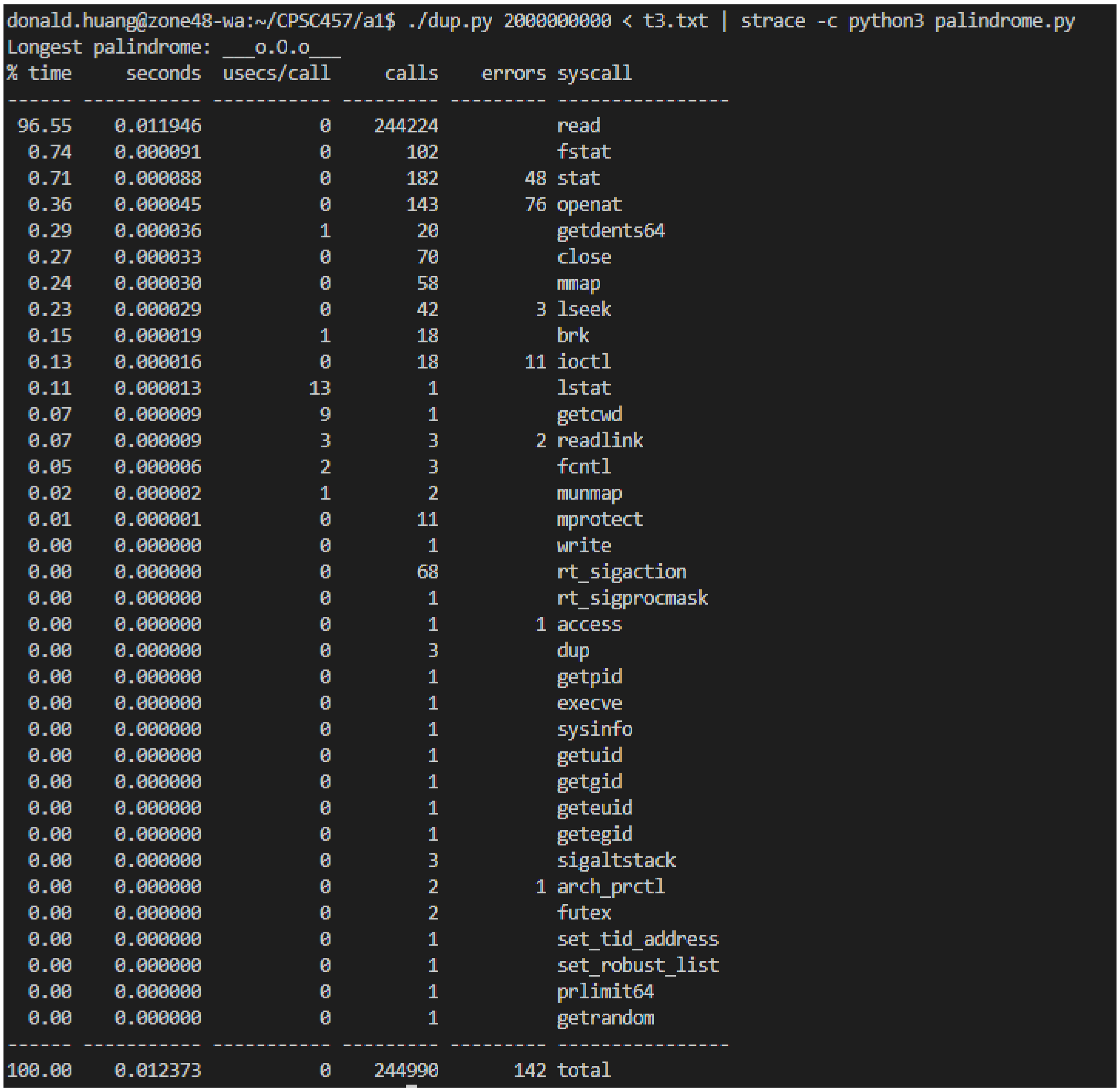
```
donald.huang@zone43-ea:~/CPSC457/a1$ time ./slow-pali < t2.txt
Longest palindrome: Bob

real    0m0.004s
user    0m0.001s
sys     0m0.002s
donald.huang@zone43-ea:~/CPSC457/a1$ time ./fast-pali < t2.txt
Longest palindrome: Bob

real    0m0.002s
user    0m0.002s
sys     0m0.000s
```

We see that fast pali has better times.

3.b) Yes the c++ code is faster than the python one, this is because c++ is a faster language and its making more optimized system calls, below are pictures of strace with dup on fast-pali.cpp and palindrome.py. In the pictures below we can see that fast-pali makes a total of 3337 calls while palindrome.py makes a total of 24499.



```
donald.huang@zone48-wa:~/CPSC457/a1$ ./dup.py 2000000000 < t3.txt | strace -c ./fast-pali
Longest palindrome: ___o.O.o___
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 99.80    0.043253          13      3225           read
  0.08    0.000035           5         7           mprotect
  0.07    0.000029           1        22           mmap
  0.03    0.000011          11         1           munmap
  0.01    0.000006           2         3           brk
  0.01    0.000003           0         5           close
  0.00    0.000002           1         2         1 arch_prctl
  0.00    0.000000           0         1           write
  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         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0        48        43 openat
------ ----------- ----------- --------- --------- ----------------
100.00    0.043339          12      3337        52 total
```



```
donald.huang@zone48-wa:~/CPSC457/a1$ ./dup.py 2000000000 < t3.txt | strace -c python3 palindrome.py
Longest palindrome: ___o.O.o___
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 96.55    0.011946           0    244224           read
  0.74    0.000091           0       102           fstat
  0.71    0.000088           0       182        48 stat
  0.36    0.000045           0       143        76 openat
  0.29    0.000036           1        20           getdents64
  0.27    0.000033           0        70           close
  0.24    0.000030           0        58           mmap
  0.23    0.000029           0        42         3 lseek
  0.15    0.000019           1        18           brk
  0.13    0.000016           0        18        11 ioctl
  0.11    0.000013          13         1           lstat
  0.07    0.000009           9         1           getcwd
  0.07    0.000009           3         3         2 readlink
  0.05    0.000006           2         3           fcntl
  0.02    0.000002           1         2           munmap
  0.01    0.000001           0        11           mprotect
  0.00    0.000000           0         1           write
  0.00    0.000000           0        68           rt_sigaction
  0.00    0.000000           0         1           rt_sigprocmask
  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         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    0.012373           0    244990       142 total
```