Assignment 1

Name: Ayman Shahriar

UCID: 10180260

Tutorial: 03

Question 1:

a)

Output of timing slow-pali.cpp code with t3.txt:

```
Longest palindrome: ___o.O.o___
real    0m0.009s
user    0m0.004s
sys     0m0.002s
```

Output of timing slow-pali.cpp code with t4.txt:

```
Longest palindrome: redder
real    0m2.646s
user    0m1.234s
sys     0m1.407s
```

Output of timing palindrome.py code with t3.txt:

```
Longest palindrome: ___o.O.o___
real    0m0.020s
user    0m0.013s
sys     0m0.003s
```

Output of timing palindrome.py code with t4.txt:

```
Longest palindrome: redder
real    0m0.413s
user    0m0.254s
sys     0m0.005s
```

b) When the python code is running t3.txt or t4.txt, the cpu spends most of the time in user mode and very little in kernel mode.
In the c++ code, the cpu spends almost half the time in kernel mode for input t4.txt and spends around one-third of the time in kernel mode for input t3.txt.
So the c++ code spends much more time in kernel mode than the python code.

c) Slow-pali issues a system call for every byte read from stdin, while palindrome.py issues a system call for every line read.
So as input gets larger, the number of system calls the c++ program makes becomes much larger than the number of system calls the python program makes. Note that the more system calls a program makes, the longer it will take to execute.

Also, C++ (a compiled language) is generally much faster than python (an interpreted language),
So for small inputs, even though the c++ program makes more system calls, it will still run faster than the python program since the difference in system calls is not yet big enough. However, for larger inputs the difference in system calls get larger, and the python program starts to run faster than the c++ program.

Question 3:

Using strace on slow-pali.cpp with t4.txt as input:

% time	seconds	usecs/call	calls	errors	syscall
100.00	5.862290	2	2872239		read
0.00	0.000085	1	48	43	openat
0.00	0.000062	2	22		mmap
0.00	0.000047	6	7		mprotect
0.00	0.000017	17	1		munmap
0.00	0.000007	1	5		close
0.00	0.000006	2	3		brk
0.00	0.000005	1	5		fstat
0.00	0.000004	2	2	1	arch_prctl
0.00	0.000000	0	8	7	stat
0.00	0.000000	0	7		lseek
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		execve
100.00	5.862523	2	2872349	52	total

Using strace of palindrome.py with t4.txt as input:

% time	seconds	usecs/call	calls	errors	syscall
23.13	0.000108	0	141	76	openat
15.20	0.000071	1	59		mmap
12.85	0.000060	0	100		fstat
10.71	0.000050	0	68		close
10.49	0.000049	0	177	49	stat
8.35	0.000039	0	788		read
5.78	0.000027	2	11		mprotect
2.36	0.000011	0	42	2	lseek
1.93	0.000009	3	3		munmap
1.93	0.000009	0	66		brk
1.71	0.000008	0	18	11	ioctl
0.86	0.000004	0	68		rt_sigaction
0.64	0.000003	1	3		dup
0.64	0.000003	1	3	2	readlink
0.64	0.000003	1	2		futex
0.64	0.000003	3	1		set_robust_list
0.43	0.000002	2	1		rt_sigprocmask
0.43	0.000002	1	2	1	arch_prctl
0.43	0.000002	2	1		set_tid_address
0.43	0.000002	2	1		prlimit64
0.43	0.000002	2	1		getrandom
0.00	0.000000	0	1		write
0.00	0.000000	0	1		lstat
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	3		fcntl
0.00	0.000000	0	1		getcwd
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	16		getdents64
100.00	0.000467	0	1589	142	total

Using strace on fast-pali.cpp with t4.txt as input:

% time	seconds	usecs/call	calls	errors	syscall
0.00	0.000000	0	18		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	130	52	total

3a) Yes, our program from Q2 is much faster than slow-pali.cpp. That is because slow-pali issues a system call for every byte read from stdin, while fast-pali.cpp issues a system call for every MiB read. For example, if we trace the system calls made by these two programs on input file t4.txt, we see that slow-pali made 5,767,205 system calls just to read from the file, while fast-pali only used 18.
As mentioned in Q1, the more system calls you have, the slower your program is going to be. And since fast-pali.cpp uses much less system calls than slow-pali.cpp, it is faster.

b) Our program fast-palli.cpp is faster than palindrome.py. This is because of two reasons:

1) C++ (a compiled language) is generally much faster than python (an interpreted language).
2) As we can see from the results of `strace` above, fast-pali.cpp uses less system calls than palindrome.py. For example, with input t4.txt fast-pali.cpp uses 130 system calls while palindrome.py uses 1589 system calls.