```
/* CPSC 457 (Winter 2019)
 * Week 4 - Section 1
 * Tutorial
 * Sina Keshvadi
 *
 * Notes: No error handling!
 */
=================================================================================
write this code to sum up 1 to n.
run code by:
time ./a.out 1,000,000,000

#include <iostream>

using namespace std;

int main( int argc, char ** argv)
{
unsigned long number = atol(argv[1]);
long sum = 0;

for (long i=0; i<=number; i++)
sum+=i;

cout<<"Sum = "<<sum<<endl;

return(0);
}
=================================================================================
Now, we try to re-write the above example with thread and compare the time.

pthread_create() function to create a thread.
pthread_create() takes 4 arguments, namely: thread, attr, start, arg.
The first argument is a pointer to thread_id which is set by this function.
the second is Null (If attr is NULL, the default attributes are used).
The third argument is name of function to be executed for the thread to be created.
The fourth argument is used to pass arguments to thread.
============================================================================
// Sample thread structure
#include <pthread.h>
#include <iostream>
#include <cstdlib>
#include <unistd.h>   // for sleep
using namespace std;

void *print_hello_world(void *unused)
{
   cout<<"Hello World\n";
   return NULL;
}

int main(int argc, char *argv[])
{
  pthread_t mythread;
  cout<<"Main here ...\n";
  pthread_create(&mythread, NULL, &print_hello_world, NULL);
  sleep(1);
  return 0;
}

// compile thread codes by : g++ -lpthread a.cpp
// What's happen if you remove sleep(1); line?
============================================================================
//re-write the sum example by using threads.
```

```
// run by time ./a.out 1,000,000,000, then compare run times.

#include <pthread.h>
#include <iostream>
#include <cstdlib>
using namespace std;

unsigned long sum;

void *adder(void * number)
{
    cout << "Thread ID = " << pthread_self() << endl;

    unsigned long num = (unsigned long) number;
    for (int i = 1; i <= num; i++)
    {
        sum += i;
    }

    pthread_exit(0);
}

int main(int argc, char *argv[])
{
    unsigned long number = atol(argv[1]);
    pthread_t tid;
    pthread_create(&tid,NULL,adder,(void *) (number/2));

long sum2=0;
for (long i=(number/2)+1; i<=number; i++)
 sum2+=i;


    // Wait for the thread to exit
    pthread_join(tid,NULL);

    cout << "Sum (" << 1 << ", " << number << ") = " << sum+sum2 << endl;

    return 0;
}

==============================================================================
try the follwoing bash codes:

// find txt files in home directory
find ~/ -type f -name '*.txt'

// find txt files in current directory
find $(pwd) -type f -name '*.txt'
or
find . -type f -name '*.txt'

// find, Sort and select top 3 based on size
find $(pwd) -name "*.txt"  -printf '%s %p\n'| sort -nr | head -3
==============================================================================
- Create a txt file by nano numbers.txt
- insert 10 integers and exit

// AWK for sum numbers
awk '{total += $1} END {print total}' numbers.txt
==============================================================================
// Using System() command in cpp code

#include <iostream>
```

```
#include <cstdlib>
#include <cstring>

using namespace std;

int main()
{
cout<<"Hi!"<<endl;

system("ls");

cout<<"BYE!"<<endl;
return 0;
}
==============================================================================
// run bash file from c++ program
#include <iostream>
#include <cstdlib>
#include <cstring>

using namespace std;

int main()
{
cout<<"Hi!"<<endl;

system("./a.sh");

cout<<"BYE!"<<endl;
return 0;
}
==============================================================================
```

For assignment:
For your second question - the C++ program must behave exactly like the bash script.
That means it must accept parameters from the command line.

```
==============================================================================
// Read input argumants from command line

#include <iostream>
#include <cstdlib>
#include <cstring>

using namespace std;

int main( int argc, char ** argv)
{
// handle command line arguments
  if( argc != 3){
     cout << "Usage: <suffix> <number of files>" <<endl;
     exit(-1);
   }

string filetype = argv[1];
int number = stoi(argv[2]);

cout<<filetype<<" "<<number<<endl;

return 0;
}
==============================================================================
// Using popen()
// save your code as popen.cpp

#include <iostream>
```

```
#include <cstdio>
#define MAX_LEN 100

using namespace std;

int main()
{
   FILE *fp;
   char buffer[MAX_LEN];

//Gives the first 5 lines of the your source code
   fp = popen("cat popen.cpp | head -n 5", "r");

   while(fgets(buffer, MAX_LEN, fp))
   {
      printf("%s", buffer);
   }
   pclose(fp);
   return 0;
}
==============================================================================
strace command
for fun, profit, and debugging.
strace can be seen as a light weight debugger. It allows a programmer / user to
quickly find out how a program is interacting with the OS. It does this by monitoring
system calls and signals.

strace ./a.out
man strace
==============================================================================
```