CS575:Parallel Programming
Monte-Carlo Simulation

Student Name: Ping-Jui Liao
Student Email: liaop@oregonstate.edu
Project Number: 1
Project Name: Monte-Carlo Simulation

I do monte-carlo simulation with
NUMT (Number Of Threads) : 1 2 4 8 threads
NUMTRIALS (Number Of Trials) : 100000 1000000 5000000 100000000 trials

I implemented this by editing .cpp file and using bash script to run multiple times.

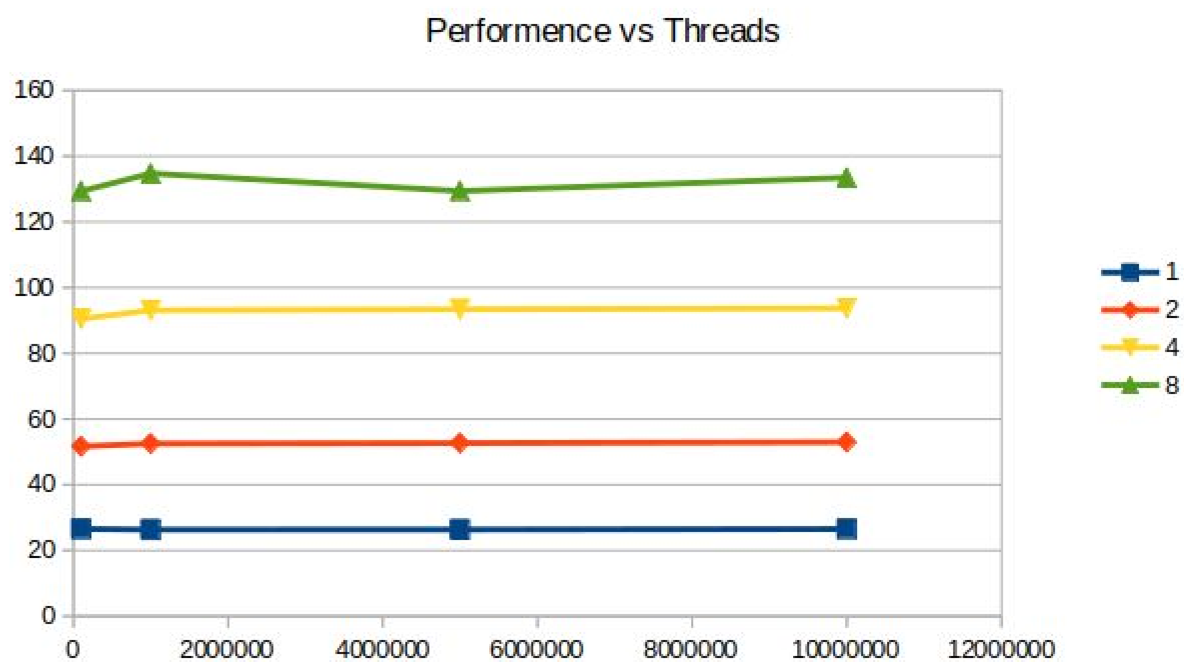
Do a table and two graphs showing performance versus trials and threads.
Table :

	100000	1000000	5000000	10000000
1	26.5427	26.3447	26.3502	26.5324
2	51.6855	52.5352	52.6884	52.953
4	90.5054	93.1108	93.3788	93.6819
8	129.361	134.743	129.383	133.456

Comparison Between Different NUMTRIALS
Color : Number of Trails
Y axis: Performance
X axis: Number of Threads

The Number of Trials do not have much influences on performances
But as the threads number goes up the differences goes upquitu

Comparison Between Different NUMTRIALS
Color : Number of Threads
Y axis: Performance
X axis: Number of Trails



As the thread number increases, the performance are significantly better.

Chosing one of the runs (the one with the maximum number of trials would be good), tell me what you think the actual probability is.

The probability under (#trial:1000000, #threads: 8)   is
0.190442

I think this is close enough to the probability based on the fact that many students get this value.

Compute Fp, the Parallel Fraction, for this computation.

Calculating the Fp = (4./3.)*(1. - (1./Speedup));

Speedup(8 thread vs. 1 thread, NUMTRIALS=1,000,000) = P8/ P1

= 134.743 / 25.3447

= 5.316417239107189

Fp = 8/7 * (1. - (1. / 5.31641..))

= **0.91940784626**

```
>>>
>>>
>>>
>>> numt = 8.0
>>> speedup = 134.743 / 26.3447
>>> print speedup
5.11461508387
>>> fp = ( numt / (numt - 1)) * (1. - (1./speedup))
>>> print fp
0.91940784626
>>>
>>>
```