Team Project1 (Deadline: 3월 30일 밤12시) :

Let's Measure the efficiency of the Prime Number Computation!

`clock()`: floating point number Return the CPU time or real time since the first call to clock()

`time()`: floating point number Return the current time in seconds since the Epoch.
The *epoch* is the point where the time starts. On January 1st of that year, at 0 hours, the "time since the epoch" is zero. For Unix, the epoch is 1970.

```
>>> import time
>>> start = time.clock()
>>> your_function()
>>> end = time.clock()
>>> duration = end – start
```

```
>>> import time
>>> start = time.time()
>>> your_function()
>>> end = time.time()
>>> duration = end – start
```

- 팀별로 Project Report를 PPT로 만들어서 제출한다
  - PPT는 intuitive, informative, visual하게 만드는것이 중요
- Deadline전에 1조_Project1.ppt 같은 화일이름 형태로 제출한다
- Deadline이 넘으면 late submission을 받지 않는다

- **Part(A)**
  - 수업시간에 배웠던 26page에 있는 sieve()와 28page에 있는 sieve()를 성능평가로 비교하는 Mission
    - 28page에 있는 sieve()를 better_sieve()로 이름을 바꾸어서 진행한다
  - Prime Number Generation의 성능평가는 아래 3경우로 한다
    - (1) 100까지의 Prime Number를 생성
    - (2) 1000까지의 Prime Number를 생성
    - (3) 10000까지의 Prime Number를 생성
  - 각 Algorithm의 processing time을 time library를 이용하여 측정해본다

- **Part(B)**
  - 수업시간에 배웠던 IsPrime_dumb(), IsPrime_better(), IsPrime_best() 를 성능평가로 비교하는 Mission
  - Prime Number Checking의 성능평가는 아래 3경우로 한다
    - (1) 1-100까지의 모든수에 대해 Prime Number여부를 test
    - (2) 1-500까지의 모든수에 대해 Prime Number여부를 test
    - (3) 1-1000까지의 모든수에 대해 Prime Number여부를 test
  - 각 Algorithm의 processing time을 time library를 이용하여 측정해본다
  - 각 Algorithm에서 수행되는 % operation의 개수를 count해본다