COM 5336 ASSIGNMENT #3

DUE BY 11:59PM **5/21**/2018 (Mon)

10% penalty applies to 1-day late submissions received between 12:00AM 5/22 and 11:59PM 5/22. No submission will be accepted after 12:00AM 5/23/2017.

Objective

Implement the Miller-Rabin primality test and the Rabin Public-Key Cryptosystem.

Description

There are 3 things you need to do before implementing the Rabin public-key cryptosystem.

(1) Big number arithmetic: Already done in assignment #1.
(2) Miller-Rabin primality test
(3) 256-bit prime number generation

Prime number generation involves pseudorandom number generation. Here you can use any pseudorandom number generating functions provided in any library (such as `srand()` in `<cstdlib>`) to do this assignment. Remember, in practice, pseudorandom number generators need to be cryptographically secure and `srand()` is NOT a good choice.

Miller-Rabin as well as any other primality tests are EXPENSIVE, so it is better to avoid running it as much as possible. You can significantly reduce your program's running time if you DO TRIAL DIVISION FOR SMALL PRIMES. IT IS STRONGLY RECOMMENDED. Below is a list of small primes up to 1000.

2	3	5	7	11	13	17	19	23	29	31
37	41	43	47	53	59	61	67	71	73	79
83	89	97	101	103	107	109	113	127	131	137
139	149	151	157	163	167	173	179	181	191	193
197	199	211	223	227	229	233	239	241	251	257
263	269	271	277	281	283	293	307	311	313	317
331	337	347	349	353	359	367	373	379	383	389
397	401	409	419	421	431	433	439	443	449	457
461	463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599	601
607	613	617	619	631	641	643	647	653	659	661
673	677	683	691	701	709	719	727	733	739	743
751	757	761	769	773	787	797	809	811	821	823
827	829	839	853	857	859	863	877	881	883	887
907	911	919	929	937	941	947	953	967	971	977
983	991	997								

Implement Rabin Cryptosystem. Users are asked to input two 128-bit primes p and q, and a 224-bit plaintext in hex. Do 16-bit repetition padding at the end as described in the class. For decryption, users are asked to input the ciphertext as well as p and q (**assumed to be either 3 mod 4 or 5 mod 8**, you don't need to deal with the case where p, q do not satisfy these requirements). Sample I/O is shown below.

output your 256-bit prime number

Sample I/O (`Input shown in` **`bold face`**.)

```
<Miller-Rabin>
xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx
xxxxxxxx

<Rabin Encryption>
p = daaefe65 2cad1614 f17e87f2 cd80973f
q = f9998862 6723eef2 a54ed484 dfa735c7
n = pq = d5375c87 792a4ac9 135966b6 d1689939 c249ed22 452f77d6
3fa82d67 e95e9cf9

Plaintext: be000bad bebadbad bad00deb deadface deafbeef add00add
bed00bed
Ciphertext = 205651dd a3fced3e 74e9c50a 61342e29 b6b8e14e 85ce5666
7b341c78 cc2965cb
```

```
<Rabin Decryption>
Ciphertext = 5452361a db4c34be 04a5903a e00793bc 1086e887 ebed06e2
3ffba0b4 a4348cc0
Private Key:
p = d5e68b2b 5855059a d1a80dd6 c5dc03eb
q = c96c6afc 57ce0f53 396d3b32 049fe2d3
Plaintext = 00000000 12345678 87654321 12345678 87654321 12345678
87654321
```

Grading

Your program MUST BE compatible with Dev C/C++ or GNU C/C++ compilers. If you are using other compilers, please make sure your final program is compatible. **You will get no points if your program is not compilable using the abovementioned compilers.** If your program is compilable but the result is not completely correct, you'll still get partial credits. Your program should be well-commented, well-structured, and easy to understand. You may lose up to 30% of points if you fail to do so.

Submission

Put all your source codes in a folder containing main functions, function implementations, class definitions, or compilation instructions, if any. Compress them as a single zip file. DO NOT submit executable files. Name your zip file as your student ID number (i.e. 100012345.zip). Submit your source code on iLMS at http://lms.nthu.edu.tw.