# CS 5973 Cryptography Project

Write a C or C++ program for the modular exponentiation. Given a prime $p$ ( with up to 30 decimal digits), a integer $n$ (with up to 300 decimal digits, maybe negative) and two polynomials $f, g \in \mathbf{F}_p[x]$(with degree 10 or less), the program should compute $g^n \bmod f$.

1. You should read the input from an ascii file with the format similiar to:

```
p = 81358375357395739583957 3993;
n = 18739277038847939886754019920358123424308469030992781
55796690998321191096315776367872612015446903085680773058 7
97185991037906908769311905108513956621737063508338494361 3
86802954525689711799860815684369946509329376583314130952 6
69635714260086693568948377087781501446119483769222387990 5
132001;
f = x^10 + 2375939583985 * x^9 + 29297593753957935395839;
g = x^8 + 92353098540385 * x^6 + 92935999999;
```

2. The output for the above example is

```
29644691226012708932131542 5 * x^9 + 98131338837641462701411 94
* x^8 + 56989506449655821889070611 3 * x^7 +
11924856390178380317028127 9 * x^6 + 12030905346373084280993728 8
* x^5 + 59056101614109495770864071 3 * x^4 +
51840856324590366491333821 4 * x^3 + 63684291925328361016646545 4
* x^2 + 61656005635436768964506379 1 * x
+ 75668818321735132685104389 5
```

   You can use this example to check whether your program is right or not.

3. I will compile your program using gcc(g++) in a UNIX system, you will not get credit if there is any compiling error. In order to get the full credit, your program must output result within 1 min.

4. You must write every function you call, except the basic input/output functions such as those described in `stdio.h`, `string.h` or `iostream.h`.

5. The assignment is due 5pm March 15(Friday). Send your program to qcheng@ou.edu. If you have two or more files, please tar them together and write the make file.