CMSC-MATH-ENEE 456 Untimed Final, Fall 2021
Morally Due Tuesday Dec 7
Dead-Cat Day is Dec 9
WARNING: TREAT Dec 7 as the DUE DATE
WARNING: DO NOT say *I did it by Thursday Dec 9, see my time stamp!*
WARNING: DO NOT say *I handed it in at "12:31PM on THURSDAY Dec 9"*
Since this is a final I am reminding you of all of this, though it has always been the case

1. This is an open-book, open-slides, open-web exam.
2. There are 4 problems which add up to 50 points.
3. In order to be eligible for as much partial credit as possible, show all of your work for each problem, **write legibly**, and **clearly indicate** your answers. Credit **cannot** be given for illegible answers.
4. Please write out the following statement: "*I pledge on my honor that I will not give or receive any unauthorized assistance on this examination.*"
5. Fill in the following:

NAME :
SIGNATURE :
UID :

1. (a) (0 points) What is the day and time of the timed final?
   (b) (0 points) Fill out the course evalutions for all of your courses.

GOTO NEXT PAGE

2. (15 points) In this problem you will end up with a program for the low-e attack if Zelda is sending a message to three people.

   Write programs for the following:

   (a) (0 points but you will use it later, so you should test it yourself.) Program CRT: Given $c_1, N_1, c_2, N_2, c_3, N_3$ find x such that

   $$x \equiv c_1 \pmod{N_1}$$
   $$x \equiv c_2 \pmod{N_2}$$
   $$x \equiv c_3 \pmod{N_3}$$

   (b) (0 points but you will use it later, so you should test it yourself.) Program NATURAL-CUBE-ROOT: Given $x \in \mathbb{N}$ output

   - 0 if $x^{1/3} \notin \mathbb{N}$.
   - $x^{1/3}$ if $x^{1/3} \in \mathbb{N}$.

   You will NEED to account for floating point errors when analyzing the cube root.

   (c) (15 points) Zelda is going to send messages to both Alice and Bob and Carol. They are using RSA with $e = 3$ (Why? Because they have not had this course.) Write a program to help Eve decode the messages using the low-e attack. More precisely:

   Program LOW-E: Input is c_1, N_1 and c_2, N_2 and c_3, N_3. Eve knows that there exists m such that the following all hold:
   $m^3 \equiv c_1 \pmod{N_1}$
   $m^3 \equiv c_2 \pmod{N_2}$
   $m^3 \equiv c_3 \pmod{N_3}$.
   Output is either

   - 0 if from this information the low-e attack won't work.
   - m if from this information the low-e attack works.

   Sample input/output:

   - Input: $c_1 = 330, N_1 = 377, c_2 = 34, N_2 = 391, c_3 = 419, N_3 = 589$
   - Output: 102

   GOTO NEXT PAGE FOR HOW TO SUBMIT

In your main method, you should take as input $c_1, N_1, c_2, N_2, c_3, N_3$ and output LOW-E$(c_1, N_1, c_2, N_2, c_3, N_3)$, which should be m or 0.

(a) Your input $(c_1, N_1, c_2, N_2, c_3, N_3)$ will be given as command line arguments, in that order. Expect your filename to be the first argument at index 0, c_1 to be the second at index 1, etc.

There is no input read through standard input.

(b) Your output should be printed to standard output. This should be an integer (ex: "5" instead of "5.0"), and this integer should be the ONLY thing printed to stdout.

(c) You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.

GOTO NEXT PAGE

3. (15 points) This is a programming assignment. You will do the Same-N attack in the case of Zelda sending to two people.

   (a) (0 points but you will need it for the other parts) Program FIND-X: On input $e_1, e_2 \in \mathbb{N}$ find $x_1, x_2 \in \mathbb{Z}$ such that

   $$e_1 x_1 + e_2 x_2 = d$$

   where d is the GCD of e_1, e_2.

   (b) (0 points but you will it for the other parts) Program MOD-INV: On input $e, N \in \mathbb{N}$ where gcd(e, N)=1, find $e^{-1} \pmod{N}$.

   (c) (15 points) Program SAME-N: On input $e_1, e_2, N, m^{e_1}, m^{e_2}$:

      i. If $e_1 \le 0$ or $e_2 \le 0$, output ”BAD INPUT: e <= 0”
      ii. If $e_1 \ge N$ or $e_2 \ge N$, output ”BAD INPUT: e >= N”
      iii. If gcd$(e_1, e_2) \ne 1$, output ”BAD INPUT: e1 NOT REL PRIME TO e2”
      iv. Otherwise, compute and output m using the Same-N attack.

   Sample Input/Output:

   - Input: $e_1 = 341, e_2 = 408, N = 1147, m^{e_1} = 883, m^{e_2} = 655$
   - Output: 15

GOTO NEXT PAGE FOR HOW TO SUBMIT

In your main method, you should take as input $e_1, e_2, N, m^{e_1}, m^{e_2}$ and output SAME-N$(e_1, e_2, N, m^{e_1}, m^{e_2})$, which should be m or one of the bad input messages.

(a) Your input $(e_1, e_2, N, m^{e_1}, m^{e_2})$ will be given as command line arguments, in that order. Expect your filename to be the first argument at index 0, e_1 to be the second at index 1, etc.

There is no input read through standard input.

(b) Your output should be printed to standard output. This should be an error message or an integer, not a float (ex: "5" instead of "5.0"). This error or m should be the ONLY thing printed to stdout.

(c) You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.

GOTO NEXT PAGE

4. (20 points) In this problem you will submit two programs, one for Alice and one for Bob, to use for the private-LWE cipher. *Notation* Whenever we say $\frac{a}{b}$ we mean $\left\lfloor \frac{a}{b} \right\rfloor$.

   (a) (0 points but you will need it) Write a program CHECK-INPUT that on input $(n, p, [r_1, \ldots, r_n], [k_1, \ldots, k_n])$ does the following

      i. If there is an i such that $r_i \leq 0$ or $r_i \geq p$ then output: "BAD INPUT: one of the ri is bad!"

      ii. If there is an i such that $k_i \leq 0$ or $k_i \geq p$ then output: "BAD INPUT: one of the ki is bad!"

      iii. Test if p is a prime. If p is NOT a prime then output: "BAD INPUT: p is not a prime you pathetic pastry!"

      iv. If $p \leq 50$ then output: "BAD INPUT: p is too small and you are small-minded!"

   (b) (10 points) (The program here is what Alice does to send Bob a bit, assuming they both have $k_1, \ldots, k_n$.) Write a program ENCRYPT-BIT that on input $(n, p, b, [r_1, \ldots, r_n], [k_1, \ldots, k_n])$ does the following

      i. Run CHECK-INPUT on $(n, p, [r_1, \ldots, r_n], [k_1, \ldots, k_n])$. If it says BAD INPUT then output whatever it output and stop.

      ii. If $b \notin \{0, 1\}$ then output: "BAD INPUT: b is not a bit you bogus bagel!" and stop.

      iii. Compute $C = \sum_{i=1}^{n} r_i k_i \pmod{p}$.

      iv. Pick a random $e \in \{-\gamma, -\gamma + 1, \ldots, \gamma\}$ where $\gamma = \left\lfloor \frac{p}{16} \right\rfloor$.

      v. Compute $D = C + e + \left\lfloor \frac{bp}{4} \right\rfloor \pmod{p}$.

      vi. Output D. Note that in the real world, Alice would also have to send Bob $[r_1, \ldots, r_n]$, which was a random vector Alice came up with.

   Sample Input/Output:

   - Input: $n = 4, p = 53, b = 1, r = [14, 23, 3, 46], k = [33, 12, 9, 16]$
   - Output: Any single value for $D \in \{27, 28, ..., 33\}$, so perhaps "31".

GOTO NEXT PAGE FOR HOW TO SUBMIT PART B

In your main method, you should take as input $n, p, b, r_1, \ldots, r_n, k_1, \ldots, k_n$ and output ENCRYPT-BIT$(n, p, b, [r_1, \ldots, r_n], [k_1, \ldots, k_n])$, which should be D or one of the bad input messages.

i. Your input $(n, p, b, r_1, \ldots, r_n, k_1, \ldots, k_n)$ will be given as command line arguments, in that order. Expect your filename to be the first argument at index 0, n to be the second at index 1, p to be the third at index 2, etc.
Notice we have a **variable amount of command line arguments**, based on n.
There is no input read through standard input.

ii. Your output should be printed to standard output. This should be an error message or an integer, not a float (ex: "5" instead of "5.0"). This error or D should be the ONLY thing printed to stdout.

iii. You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.

GOTO NEXT PAGE

(c) (10 points) (The program here is what Bob does to determine the bit sent from Alice, assuming they both have $k_1, \ldots, k_n$.) Write a program DECRYPT-BIT that on input $(n, p, [r_1, \ldots, r_n], [k_1, \ldots, k_n], D)$ will do the following:

i. Run CHECK-INPUT on $(n, p, [r_1, \ldots, r_n], [k_1, \ldots, k_n])$. If it says BAD INPUT then output whatever it output and stop.

ii. Output the bit b that has been encoded.

You will need to determine the range of values D can have when $b = 0$ and when $b = 1$. You can assume D has been computed properly.

Sample Input/Output:

- Input: $n = 4, p = 53, r = [14, 23, 3, 46], k = [33, 12, 9, 16], D = 31$
- Output: "1"

GOTO NEXT PAGE FOR HOW TO SUBMIT PART C

In your main method, you should take as input $n, p, r_1, \ldots, r_n, k_1, \ldots, k_n, D$ and output DECRYPT-BIT$(n, p, [r_1, \ldots, r_n], [k_1, \ldots, k_n], D)$, which should be the encoded bit b or one of the bad input messages.

(a) Your input $(n, p, r_1, \ldots, r_n, k_1, \ldots, k_n, D)$ will be given as command line arguments, in that order. Expect your filename to be the first argument at index 0, n to be the second at index 1, p to be the third at index 2, etc.

Notice we have a **variable amount of command line arguments**, based on n.

There is no input read through standard input.

(b) Your output should be printed to standard output. This should be an error message, "0", or "1", and this should be the only thing printed.

(c) You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.