CIS 3362 Homework #4: Classical Ciphers - Coding
Due: Check WebCourses for the due date.

Directions: To be done in pairs. If you can't find someone to work with, you must submit individually. When you submit, please CLEARLY mark both group members on each document you submit.

Arup's Base 8 cipher

After teaching the Playfair cipher and the ADFGVX cipher for several years, Arup was motivated to make his own cipher. The steps of the cipher are described below:

The plaintext for Arup's Base 8 cipher must be a sequence of characters with Radix-64 encodings. (If you are not familiar with this encoding system, please Google it.) This allows for all lowercase letters, uppercase letters, digits, the plus sign and the forward slash. The key for the cipher is the 64 possible plaintext characters placed in an 8 x 8 square, where each row and column are labeled 0 through 7, inclusive. Below is a possible key for the cipher. The red colored column is not part of the key and is simply the row labels and the blue colored row is also not part of the key and is simply the column labels. Also, ambiguous symbols in print are labeled with LL standing for lowercase letter, UL standing for uppercase letter and dig standing for digit.

	0	1	2	3	4	5	6	7
0	q	T	1(dig)	o(LL)	F	J	K	x
1	7	d	M	S	k	t	f	P
2	j	C	y	c	+	3	E	m
3	/	p	u	X	U	s	l (LL)	0 (dig)
4	e	I (UL)	Y	B	G	b	w	4
5	r	h	Q	6	9	W	g	Z
6	N	8	A	2	n	L	v	O(UL)
7	i	H	z	V	R	a	D	5

The second key for the cipher is a prime number, *p*.

We encrypt as follows:

1) Box encryption

We go through each letter in the plaintext of length L, in the following fashion:

for i in [0, L-1]:

a) Find the plaintext letter in the box, the corresponding cipher text is xy, where x and y are the row and column numbers, respectively of the location of the plaintext character.

b) RotateBox(i)

Here is how the RotateBox function works:

if i is even, we cyclicly rotate row (i%16)/2 to the right by one position

if i is odd, we cyclicly rotate column (i%16)/2 (int division) down by one position.

For example, after one character is encrypted, the new state of the box above is:

	0	1	2	3	4	5	6	7
0	x	q	T	1(dig)	o(LL)	F	J	K
1	7	d	M	S	k	t	f	P
2	j	C	y	c	+	3	E	m
3	/	p	u	X	U	s	l (LL)	0 (dig)
4	e	I (UL)	Y	B	G	b	w	4
5	r	h	Q	6	9	W	g	Z
6	N	8	A	2	n	L	v	O(UL)
7	i	H	z	V	R	a	D	5

After two characters are encrypted the new state of the box is:

	0	1	2	3	4	5	6	7
0	i	q	T	1(dig)	o(LL)	F	J	K
1	x	d	M	S	k	t	f	P
2	7	C	y	c	+	3	E	m
3	j	p	u	X	U	s	l (LL)	0 (dig)
4	/	I (UL)	Y	B	G	b	w	4
5	e	h	Q	6	9	W	g	Z
6	r	8	A	2	n	L	v	O(UL)
7	N	H	z	V	R	a	D	5

When this phase is done, the intermediate ciphertext should be a sequence of 2L digits in base 8 (0 through 7). Let this sequence be $d_1, d_2, ..., d_{2L}$.

Next, we use the secret key, the prime number, p, to generate a sequence of numbers as follows:

List the 2L + 1 prime numbers after p, in order. Let this sequence be $p_1, p_2, ..., p_{2L+1}$. Then, create the sequence s such that $s_i = (p_{i+1} - p_i)/2$, for all integers i in [1, 2L].

Then, calculate the 2L base 8 digits of the ciphertext $c_i = (d_i + s_i)\%8$, for all integers i in [1, 2L].

Finally, convert each pair of input (c_{2i-1}, c_{2i}) for all integers i in [1, L] to a single character in Radix-64. Namely, First convert the pair to base 10 $(8c_{2i-1} + c_{2i})$, then convert this to its corresponding Radix-64 value ('A'-'Z' is 0-25, 'a'-'z' is 26-51, '0'-'9' is 52-61, '+' is 62 and '/' is 63.)

Your Assignment

In either Python, C, C++ or Java, write a program that carries out encryption using this cipher. Your program should read in input from standard input and produce output to standard output.

Input Format

The first line of input will have a single positive integer, *n*, the number of encryptions to carry out. The first eight lines of each input case will have 8 characters each, representing the 8 x 8 input key. Thus, these lines will contain each of the Radix-64 characters exactly once. The ninth line of each input case will contain a single positive integer, *p* ($p < 10^5$), the second key for the cipher. The last line of each input case will contain a string in between 1 and 100 Radix-64 characters representing the plaintext.

Output Format

For each input case, output the corresponding plaintext (which should only consist of Radix 64 characters) on a line by itself.

Sample Input and Output

To be provided later.

Deliverables

A single source file in C, C++, Python or Java (`arupcipher.c, arupcipher.cpp, arupcipher.py or arupcipher.java`). Your source file **MUST HAVE** both group member's names in the header comment. Only one person in the group of two should submit the file. If you are working by yourself, just put one name in your header file and submit your file.

Notes

Much of the grade will be based on execution. Any program that crashes will receive a grade of less than 50%, regardless of how many cases correctly executed before crashing. The Sieve of Eratosthenes should be used to generate a large enough stream of prime numbers in a pre-computation step before reading the input to save some time. If you haven't seen this, look it up. It's very short to code.