CSE 232 Spring 2018

Programming Project 05

Assignment Overview

This assignment is worth 40 points (4.0% of the course grade) and must be completed and turned in before 11:59pm on Monday, February 26th , 2018. That's two weeks because of the midterm on ***Wednesday, Feb 14th in the evening, 7pm, 1281 ANH*** .

Background, Foursquare Cipher

A cipher is an algorithm used to send a message in such a way that only someone who understands the algorithm can understand the underlying message. The process of creating the cipher is called encoding, and recovering the original message is decoding. There are lots of interesting ciphers in the world, so much so that Simon Singh wrote a book on it, "The Code Book".

We are going to look at a particular cipher Felix Delastellle (1840-1902) wrote in a book in 1902. It has a 2D graphical explanation which we will discuss, but we can easily write an algorithm to do the work with simple linear strings.

Wikipedia does a pretty good job of explaining it, please see https://en.wikipedia.org/wiki/Four-square_cipher and follow along.

Details

There are 4 blocks in a foursquare cipher (shocking!). Let's label them 0,1,2,3.

0

	0	1	2	3	4
0	a	b	c	d	e
1	f	g	h	i	j
2	k	l	m	n	o
3	p	r	s	t	u
4	v	w	x	y	z

1

	0	1	2	3	4
0	e	x	a	m	p
1	l	b	c	d	f
2	g	h	i	j	k
3	n	o	r	s	t
4	u	v	w	y	z

3

	0	1	2	3	4
0	k	e	y	w	o
1	r	d	a	b	c
2	f	g	h	i	j
3	l	m	n	p	s
4	t	u	v	x	z

2

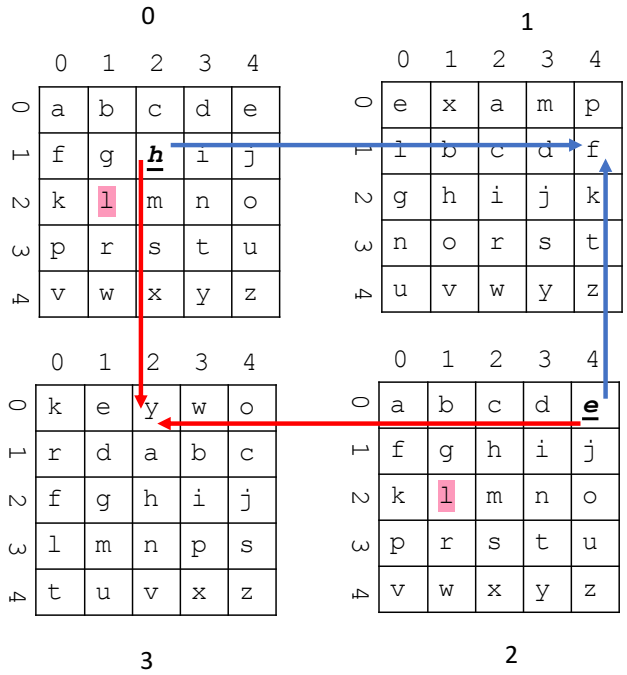
	0	1	2	3	4
0	a	b	c	d	e
1	f	g	h	i	j
2	k	l	m	n	o
3	p	r	s	t	u
4	v	w	x	y	z

You'll note that blocks 0 and 2 are the same, an enumeration in two-D of the alphabet ***minus the letter q***, thus making a 5x5 2D block. Blocks 1 and 3 are more interesting.

Keywords
Block 1 represents an encoding of the keyword "example". The letters are enumerated, starting with the keyword, so that only 1 occurrence of each letter is represented. Thus, the first letters in Block 1 are "exampl" (note the last 'e' is not included, it is already in the block). Subsequently all the letters that are not represented are place in the block ***in alphabetical order***! The same is then done for Block 2 using the keyword "keyword".

Encoding
Encoding uses the 4 blocks to encode a message in groups of two characters. Note that the underlying message is assumed to have an even number of letters, and if not the letter 'x' is added arbitrarily at the end of the message. Let's encode (as the Wikipedia does) "Hello World!"

First, we process the message so that it consists of only alphabetic, lower case letters. Thus we will actually be encoding "helloworld". Done in pairs, the first digraph (an example of two letters) is "he". Lets encode that:



We look up the two letters of "he", the first, 'h', in Block 0 and the second, 'e' in Block 2. We use their location as a way to access letters in the same row and column in Blocks 1 and 3. The 'h' and 'e' point to 'f' in Block 1 and 'y' in Block 3. The first two letters are therefore encoded as 'fy', in that order. Keep repeating that process for every pair of letters in the message, you get: "fyhghzhsje". Follow the blue arrows for encoding.

Decode
To decode is pretty simple, you reverse the process. That is, to find the original two letters encoded by "fy", you find the 'f' in Block 1 and the 'y' and Block 3, then see what they point to in Block 0 and Block 2. Follow the red arrows for decoding

Representing a 2D block as a string

1

	0	1	2	3	4
0	e	x	a	m	p
1	l	b	c	d	f
2	g	h	i	j	k
3	n	o	r	s	t
4	u	v	w	y	z

It seems like you would need a 2D vector or array to do what we just described, but you really don't. You have to realize the following relationship between a 1D string and a 2D block. Let's do it with our Block 1 from above encoded with "example". That block is show again. However, you might want to look at the relationship between the 1D string `"examplbcdfghijknorstuvwyz"` and that 2D block. Let's look at the letter 'h'. The 'h' in this string occurs at index 11 (remember, first index is 0) in the 1D string, or at row 2, column 1 of the 2D block. In the block, each row occupies 5 elements, and there are 5 rows. The relationship between the 1D index and the 2D row/col values are:

row = index / 5 (using integer division).	11 / 5 → 2	/ 5 because there are 5 rows
col = index % 5 (modulus)	11 % 5 → 1	% 5 because 5 elements/row
index = row * 5 + col	2 * 5 + 1 → 11	

Thus, we can find the equivalent 2D row/col values if we have a 1D index in a string, and if we have the 2D row/col values we can convert that to a 1D index in a string. That's all we need.

Your Tasks

Complete the Project 5 by writing code for the following functions. Details of type for the functions can be found in `proj05_functions.h` (provided for you, see details below):

Functions

string clean_string(string s)

- returns a new string that is all lowercase and consists of only alphabetic characters in s
- If s is "Hello World!", clean_string returns "helloworld"

string create_encoding(string key)

- takes in a key, and returns a keyword encoding as shown above
- `"example"` returns `"examplbcdfghijknorstuvwyz"`

string encode_digraph(string dg, string block1, string block2)

- takes in the two keyword blocks as encoded by create_encoding above
- converts the digraph string dg to its encoded digraph
- encode_digraph("he", "example", "keyword") returns "fy"

string decode_digraph(string dg, string block1, string block2)

- takes in the two keyword blocks as encoded by create_encoding above
- converts the ***encoded*** digraph string dg to its original digraph
- decode_digraph("fy", "example", "keyword") returns "he"

string encode(string msg, string key1, string key2)

- takes in the message to encode and the two keywords used for encoding

    - if the original message has an odd number of letters, 'x' is added to the end
- returns a string which is the encoding of the message

string decode(string msg, string key1, string key2)

- takes in the encoded message to decode and the two keywords used for encoding
- returns a string which is the decoding of the message

Deliverables

`proj05/proj05_functions.cpp` -- your completion of the functions described above.
Only `proj05/proj05_functions.cpp` is turned in to Mimir.

1. Remember to include your section, the date, project number and comments.
2. Please be sure to use the specified directory and file name.

Separate compilation is a topic for the next week, but most of week06 is open for those trying to get ahead. Week6's project and lab are, however, closed.

Manual Grading

In lab05 you worked with Code Style. The TAs will apply the things that you learned in your Code Review lesson on the code you submit. 4 points are reserved (out of 50) for this.

Assignment Notes

1. Mimir allows us to test the functions in `proj05_functions.cpp` individually and we will do that.
2. You can write your own `pro05_main.cpp` to test your code, but only `proj05_functions.cpp` is turned into Mimir
3. `proj05_functions.h` is provided in the project05 directory. It will be used in testing. When we do testing, we will use this file. If you change and submit your own version, Mimir will ignore it and use ours.