Cairo University, Faculty of Computers
and Artificial Intelligence

FACULTY OF COMPUTERS AND AI, CAIRO UNIVERSITY

CS112: Structured Programming
Year 2023-2024

Second Semester

Assignment 2 – List of Ciphers and Calculators V8.0

Course Instructors:

Dr. Mohammad El-Ramly

Revision History

Version 5.0	By Dr Mohammed El-Ramly 28 Feb. 2022	21/22 Ciphers Update
Version 6.0	By Dr Mohammed El-Ramly 8 Mar. 2022	Fixed Caesar Example
Version 7.0	By Dr Mohammed El- Ramly26 Feb. 2024	2023/24 Version Route instead of Caesar
Version 8.0	By Dr Mohammed El- Ramly3 Mar. 2024	Fixed Polybius & XOR

CS112: Structured Programming

Assignment 2 – Cipher and Calculator Description

Cairo University, Faculty of Computers and Artificial Intelligence

Introduction

This document describes the 10 ciphers & calculator to be coded by CS112 students in 2023/24. Make sure to implement the correct encryption/decryption algorithms according to your ID.

Cipher algorithms are very important to encrypt private data and protect them from hackers. Armies use complex algorithms to protect their communications from the enemies. Read about ciphers her https://www.crypto-it.net/eng/simple/index.html

List of Ciphers – First Cipher Number Is 0 – Choose the Right One

0. Affine Cipher

In affine cipher each letter in an alphabet is mapped to its numeric equivalent x, encrypted using a simple mathematical function, and converted back to a letter. Letter A is given number 0 and letter Z is given number 25. Each letter is encrypted with the function $(5x + 8) \bmod 26$. The decryption function is $21(y - 8) \bmod 26$. See examples at: https://cryptii.com/affine-cipher/.

Example

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Plain text	A	F	F	I	N	E	C	I	P	H	E	R
X	0	5	5	8	13	4	2	8	15	7	4	17
(5*x* + 8)	8	33	33	48	73	28	18	48	83	43	28	93
(5*x* + 8) mod 26	8	7	7	22	21	2	18	22	5	17	2	15
Cipher text	I	H	H	W	V	C	S	W	F	R	C	P

Ciphertext	I	H	H	W	V	C		S	W	F	R	C	P
y	8	7	7	22	21	2		18	22	5	17	2	15
21(y - 8)	0	-21	-21	294	273	-126		210	294	-63	189	-126	147
21(y - 8) mod 26	0	5	5	8	13	4		2	8	15	7	4	17
Plaintext	a	f	f	i	n	e		c	i	p	h	e	r

Make a general version that takes three parameters *a, b* and *c* and does the encryption and decryption according to these equations:

E(x) = (a x + b) mod 26 where x is the numeric value of the letter to cipher.

D(y) = c (y - b) mod 26 where y is the numeric value of the letter to decipher.

a, b, c are arbitrary positive integers that satisfy the condition **(a * c) mod 26 = 1**

1. Route Cipher

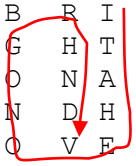
Route Cipher is one of the simplest and most widely known encryption techniques. In this cipher, a secret integer key is used to create a matrix whose number of columns is equal to the key and then the message is written in as many rows as needed in this matrix. Then the encrypted message is collected by going in a spiral path starting from the top right corner.

For example, let's encrypt a name of a city the UK, **`Brighton and Hove`**. The secret key will be 3, and it will determine the width of the matrix. We will ignore all spaces and turn all letters to

capital. We will exclude any non-letter characters. Then we will fill the matrix row by row, from left to right. If there are still empty cells in the matrix, we will them fill them with 'X'. Finally, we will read the grid clockwise, going inwards, and starting from the top right corner.

The original message will be: **BRIGHTONANDHOVE**

The letters are then entered into the grid, which is 3-column wide:



Luckily, in our case, there is no need to add any additional characters at the bottom of the grid.

The letters are then read, and appended to the cipher text. The reading starts from the top right, and spiral clockwise inwards.

Decryption involves going in the opposite process.

The produced encrypted text will be: **ITAHEVONOGBRHND**

Read more on: https://www.crypto-it.net/eng/simple/route-cipher.html

2. Atbash Cipher

The Atbash cipher is a very common, simple cipher. Basically, when encoded, an "A" becomes a "Z", "B" turns into "Y", etc. See http://rumkin.com/tools/cipher/atbash.php. Example:

```
Plain:    ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:   ZYXWVUTSRQPONMLKJIHGFEDCBA

Plain:    MOHAMMAD ELRAMLY
Cipher:   NLSZNNZW VOIZNOB
```

Make another version that divides the alphabet into 2 halves and does the same thing on each half separate. So: **Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ** & Cipher: **MLKJIHGFEDCBA ZYXWVUTSRQPON**

3. Vignere Cipher

In this method, a keyword is repeatedly added character by character to each alphabetic letter in the original message. The addition is carried out using the ASCII codes for each of the characters, modulo 26 (the number of letters in the alphabet), and the result is added to the code for the letter 'A' in the ASCII code sequence. For example, if the original message is "due November 4" and the keyword is "HWone", the message will be encrypted as follows:

```
                message:    DUE NOVEMBER 4
                            ||||||||||||||
  repeated keyword:         HWONEHWONEHWON
                            ||||||||||||||
encrypted message:          KQS RVRSZFLN 4
```

The steps used to encode the first character are:

```
'D' = ASCII 68, 'H' = ASCII 72
68+72 = 140
140%26 = 10
65 (ASCII 'A') + 10 = 75
ASCII 75 = 'K'
```

Assumptions and Restrictions. The message to be encoded and the keyword will both be read in from the keyboard. Only alphabetic characters will be encoded; all other characters will be outputted unchanged. Check to ensure keyword is only alphabetic characters. All alphabetic characters should be converted to uppercase before the encoding process begins. The input message should be restricted to 80 characters, the keyword to 8 characters (your program needs to check these limits).

CS112: Structured Programming

Assignment 2 – Cipher and Calculator Description

Cairo University, Faculty of Computers and Artificial Intelligence

4. Baconian Cipher

To encode a message, each letter of the plaintext is replaced by a group of five of the letters 'A' or 'B'. This replacement is a binary encoding. For example, SAMY will be baaba aaaaa abbaa bbaaa. When deciphering, sequences of a and b that do not correspond to a letter are ignored.

Letter	Code	Binary	Letter	Code	Binary	Letter	Code	Binary	Letter	Code	Binary
A	aaaaa	00000	G	aabba	00110	N	abbab	01101	U	babaa	10100
B	aaaab	00001	H	aabbb	00111	O	abbba	01110	V	babab	10101
C	aaaba	00010	I	abaaa	01000	P	abbbb	01111	W	babba	10110
D	aaabb	00011	J	abaab	01001	Q	baaaa	10000	X	babbb	10111
E	aabaa	00100	K	ababa	01010	R	baaab	10001	Y	bbaaa	11000
F	aabab	00101	L	ababb	01011	S	baaba	10010	Z	bbaab	11001
			M	abbaa	01100	T	baabb	10011			

See http://rumkin.com/tools/cipher/baconian.php

5. Simple Substitution Cipher.

In this cipher, a replacement alphabet is used to replace each letter by another one. See http://practicalcryptography.com/ciphers/simple-substitution-cipher/
For example, if we use this cipher alphabet:

```
plain alphabet : abcdefghijklmnopqrstuvwxyz
cipher alphabet: phqgiumeaylnofdxjkrcvstzwb
```

We can encrypt the following sentence as follows: (convert message and key to lower or upper case)

```
Plain text : I love C plus plus
Cipher text: a ndsi q xnvr xnvr
```

Create a general version that builds the cipher alphabet using a **given key of 5 unique letters**. The user enters the key to cipher a message and the same key to decipher the message. The cipher alphabet is built by adding the remaining 21 letters (excluding the 5 letters entered) in order after the key letters. For example, if the user enters "**zebra**" as the key, then:

```
plain alphabet : a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher alphabet: z e b r a c d f g h i j k l m n o p q s t u v w x y
```

The, we can encrypt the following sentence as follows:

```
Plain text : I love C plus plus
Cipher text: g jmua b njtq njtq
```

If the user enters "cairo" as the key, then:

```
plain alphabet : a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher alphabet: c a i r o b d e f g h j k l m n p q s t u v w x y z
```

The, we can encrypt the following sentence as follows:

```
Plain text : I love C plus plus
Cipher text: f jmvo i njus njus
```

CS112: Structured Programming

Assignment 2 – Cipher and Calculator Description

Cairo University, Faculty of Computers and Artificial Intelligence

6. Polybius Square Cipher

See how it works at https://en.wikipedia.org/wiki/Polybius_square and https://www.dcode.fr/polybius-cipher using numbers. Each letter is replaced with two numbers according to the following table (row, column). Write another version where top and left numbers are given as a key, e.g., instead of 1 | 2 | 3 | 4 | 5 , user can enter the key 2 | 4 | 5 | 3 | 1 but you will need to have the same key to decipher the message. So, if the sender enters the key 1 2 3 4 5

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

```
Plain text : I  love     C  plus     plus
Cipher text: 24 31345115 13 35314543 35314543
```

If the user enters the key 5 1 4 2 3, then we have

	5	1	4	2	3
5	A	B	C	D	E
1	F	G	H	I/J	K
4	L	M	N	O	P
2	Q	R	S	T	U
3	V	W	X	Y	Z

```
Plain text : I  love     C  plus     plus
Cipher text: 12 45423553 54 43452324 43452324
```

(you can ignore spaces and other non-alphabetic characters)

To decipher the message, you must enter the same key used to encrypt it. So, the sender who ciphers the message and the receiver who decipher it BOTH must know the same key. You may need to use 2D arrays. Note that I/J are encrypted to the same numbers by when decrypted, we take I since it is more common. This means that words with J will have all Js replaced by Is.

7. Morse Code

It is a code consisting of two symbols **dot** and **dash** and used to in the telegraph system in the past and also communicate messages in primitive ways. See http://www.unit-conversion.info/texttools/morse-code/

Develop a program to translate a message to Morse code and the opposite. Assume that each letter is separated by one space from the next and that each word is separated by three spaces from the next. Example:

```
Plain text: I love C plus plus
Morse text:
..□.-..□---□...-□.□-.-.□.--.□.-..□..-□...□.--.□.-..□..-□...
```

8. XOR Cipher

In this cipher, a secret key consisting of one letter (or more) is given from the sender to the reciver. Key can be any number of letters. Then each letter of the message goes through XOR operation with one of the **secret letters** in order as in example. The output is printed in **text** and **hexadecimal**. The original message can be recovered from the encrypted message by the same algorithm, XOR with the secret letter. For example, assume this encryption example:

```
Input:
Secret key is 'P'= 01010000 (in binary)
Plain text: abcdefgh_ABCDEFG (all letters are XOR with 'P')
Output:
Cipher text: 12345678        (Some characters are non-printable)
Hexa: 31 32 33 34 35 36 37 38 0f 11 12 13 14 15 16 17

Input:
```

CS112: Structured Programming

Assignment 2 – Cipher and Calculator Description

Cairo University, Faculty of Computers and Artificial Intelligence

```
Secret key is 'zZ'= 01111010 01011010 (in binary)
Plain text:  abcdefgh_ABCDEFG (1st letter XOR with 'z', 2nd with 'Z')
Output:
Cipher text: 8><2%8>< (Some characters are non-printable)
Hexa: 1b 38 19 3e 1f 3c 1d 32 25 1b 38 19 3e 1f 3c 1d
```

Assume this decryption example:

```
Input:
Secret key is 'zZ'= 01111010 01011010 (in binary)
Cipher hexa: 1b 38 19 3e 1f 3c 1d 32 25 1b 38 19 3e 1f 3c 1d
Output:
Plain text: abcdefgh_ABCDEFG
```

See http://md5decrypt.net/en/Xor/#results for trying the tool and use this to convert hexa to text https://www.rapidtables.com/convert/number/hex-to-ascii.html

Some letters combined with some keys will produce unreadable characters. **In reality, this is not a problem.** But for us, we like to be able to reenter the ciphered message. **So, you need to print the ciphered text in hexa also (**as above) and **allow the user to enter the message to decipher as hexa similar to the shown example.**

9. **Rail-fence Cipher**

   See the details at http://practicalcryptography.com/ciphers/classical-era/rail-fence/. You may remove spaces and convert all letters to small letters and assume a fixed key value, e.g., 3 or 4.

What to submit? (Group submission of code)

1- Write the **algorithms** you used in encryption and decryption in the report.
2- **Individual report** name should be **CS112_A2_T1_ YourSection_YourID1.pdf**
3- Submit **group code** plain C++ file **CS112_A2_T2_Section_YourID_ID_ID.cpp**
4- Add comments and a header like this to your code and **write clean code.**

```
// File: File Name
// Purpose: Description of your game
// Author: Your Names &  Section
// Who did which cipher
// Emails: ...
// IDs: Your IDs and who did which part
```

CS112: Structured Programming

Assignment 2 – Cipher and Calculator Description

Cairo University, Faculty of Computers
and Artificial Intelligence

Rational Number Calculator

- In this application create a rational number calculator that is capable of taking two rational numbers and an operation to perform on them. Program should handle cases of –ve numbers and nominator without denominator. It should also use **defensive programming** to reject bad inputs.

  Example program run:

  - Please enter a rational number operations (or exit):
  - 1/2 + 1/4
  - = 3/4
  - 5/3 - 3/5
  - = 14/15
  - 1 * 12/5
  - = 12/5 (or better write it as 2 2/5
  - ……
  - 3/4 / 32/3
  - = 32/4 (or better represent as 4)
  - 1ss3 + 4/3
  - Invalid operand. Try again.
  - 3 +g% 7/11
  - Invalid operation. Try again.
  - exit
  - Thank you for using rational number calculator

  (Hint: You may find functions **`stoi`** and **`getline()`** and **`regex`** library useful.)

What to submit?

1- Write the algorithms you used in encryption and decryption in the report.
2- Program name should be **CS112_A2_T3_SectionNum_ YourID1_YourID2_YourID3.cpp**
3- Explain your algorithm in pseudo-code inside the cpp file
4- Add comments and a header like this to your code and **write clean code.**

```
// File: File Name
// Purpose: Description of your game
// Author: Your Name and Your Section
// Emails: ...
// ID1: Your ID1 – the part s/he did
// ID2: Your ID2 – the part s/he did
// ID3: Your ID2 – the part s/he did
```