COMP 3350 Project #4

Vigenère Cipher

Possible points: 100 Due: April 26, 2019 4:59pm

No homework will be accepted after the due date, so make sure to start as early as you can. You might face some issues and need the GTA help.

Assignment Goal:

1. Using critical thinking and problem solving.
2. Get you familiar with :
    1. Defining and accessing Arrays.
    2. Procedures
    3. Jumps and Conditional jumps.
    4. Registers and instructions.
    5. ASCII and text manipulation.
    6. Dealing with Loops.
    7. Debugging and running your assembly code.

Deliverables:

Submit the source file (.asm) to Canvas before the due date. This should be the only file you submit to Canvas.

The file should be named {USERNAME}_P{NUMBER}.asm

USERNAME is your auburn email without "@auburn.edu"

E.g. abc0003_P4.asm

Specifications:

This program has two objectives. The first objective is to create a procedure that will take plaintext and an encryption key, use the Vigenère cipher, and encrypt the plaintext. The second objective is to create a procedure that will take cypher-text, an encryption key, use the Vigenère cipher, and decrypt the cypher-text.

Assume that all characters will be uppercase letters, no spaces, symbols, etc.

All "high level" directives are not allowed on this homework. (e.g. .IF .ENDIF .REPEAT, etc)

Design:

Create a BYTE array with the label 'input'. This array may be of any length between 2 and 100. In the case of encryption 'input' will hold the plaintext and in the case of decryption 'input' will hold cypher-text .

Create a BYTE array with the label 'key'. This array will have length between 1 and (LENGHTOF input -1). Meaning the lower bound for 'key' is one character and the upper bound is one less than the number of characters in 'input'.

Create a BYTE variable named 'options'. This variable will be used to determine which procedures should be executed. If 'options' contain the value 1 it means the program should execute encryption procedure. If 'options' contain the value 0 (or any other value than 1) the program should execute decryption procedure.

Create a BYTE array with the label 'output'. This array will have a dynamic length equal to LENGTHOF input. When executing the program using encryption, the variable 'output' will hold the cypher-text and in the case of decryption, 'output' will hold plaintext.

You may create any other values you deem necessary.

You must have three procedures for this homework. The main procedure, a procedure to handle encryption, and another to handle the decryption. You may create other procedures if you wish.

Example of encryption / decryption:

> For more information check the Wiki link: (https://en.wikipedia.org/wiki/Vigenère_cipher)

The example below will use the plaintext word MEMORY and the key BAD.

First you "line up" your plaintext word with your key by writing the key directly beneath the plaintext word. Continue to repeat the key under each plaintext character:

MEMORY

BADBAD

In the above example MEMORY is longer than BAD, therefore, BAD was repeated.

Next choose each character in the plaintext word, starting with the first and encrypt it.

To encrypt a character, you first find the column on the chart corresponding to the character in the plaintext, e.g. the character M (circled in blue). Next, find the row starting with the key character, e.g. the character B (circled in red). Finally, find where

the plaintext column and the key row intersect, e.g. the character N (circled in yellow). Continue this for each character.

MEMORY

BADBAD

N

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

MEMORY

BADBAD

NE

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

MEMORY

BADBAD

NEP

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

Continue until all characters have been encrypted:

Input = MEMORY (plaintext)

Key = BADBAD (key)

output= NEPPRB (cypher-text)

Decrypting is the opposite of encrypting. To decrypt NEPPRB first write the key under the cypher-text. Then find the row starting with the key character. Next, move across that row until you find the cypher-text character. The cypher-text character's column is the plaintext column.

NEPPRB

BADBAD

M

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

NEPPRB

BADBAD

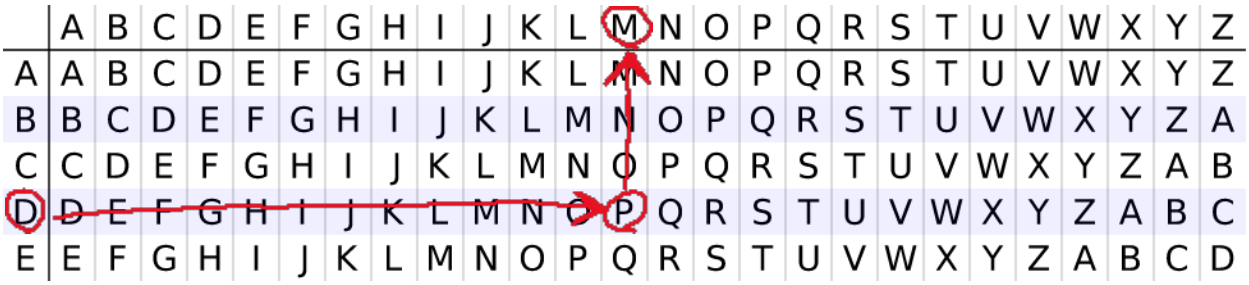
ME

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

NEPPRB

BADBAD

MEM



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

Continue until the cypher-text has been decrypted:

Input = NEPPRB

Key = BADBAD

Output = MEMORY

Code Documentation:

At the top of your "asm" file, give a brief description of the program, author name and last modified date in the form of comments.

For the code, explain what each assembly line mean in plain English. Your comment should be meaningful and reflect the code correctly.

Late Submission Penalty:

- Late submissions will not be accepted and will result in a ZERO without valid excuses, in which case you should talk to Dr. Li to explain your situation.

- GTA/Instructor will NOT accept any late submission caused by Internet latency or internet issues.