UTM
UNIVERSITI TEKNOLOGI MALAYSIA

DEPARTMENT OF COMPUTER SCIENCE & SERVICES

CENTRE FOR DIPLOMA STUDIES, SPACE

(DDWC 3343)
COMPUTER SECURITY
ASSIGNMENT 1
VERNAM CIPHER AND CAESER CIPHER IMPLMENTATION IN C++

LECTURER NAME AND SECTION

NUR AINI BINTI ZAKARIA
SECTION 38

STUDENT NAME AND MATRIC ID

AHMAD FARIS AIMAN BIN ARIZAL
A18DW0052

ADAMM ILMAN BIN SUHAIMI
A18DW0187

AHMAD DANIAL HARITH BIN AHMAD KAMAL
A18DW1100

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

01

TABLE OF CONTENTS

Contents

Team Members

Team members tasks are divided in the following manner.

Name	Matric ID	Role	Task
ADAM ILMAN BIN SUHAIMI	A18DW0187	Programmer	Ceaser Cipher
AHMAD DANIAL HARITH BIN AHMAD KAMAL	A18DW1100	Programmer	Ceaser Cipher
AHMAD FARIS AIMAN BIN ARIZAL	A18DW0052	Programmer	Vernam Cipher

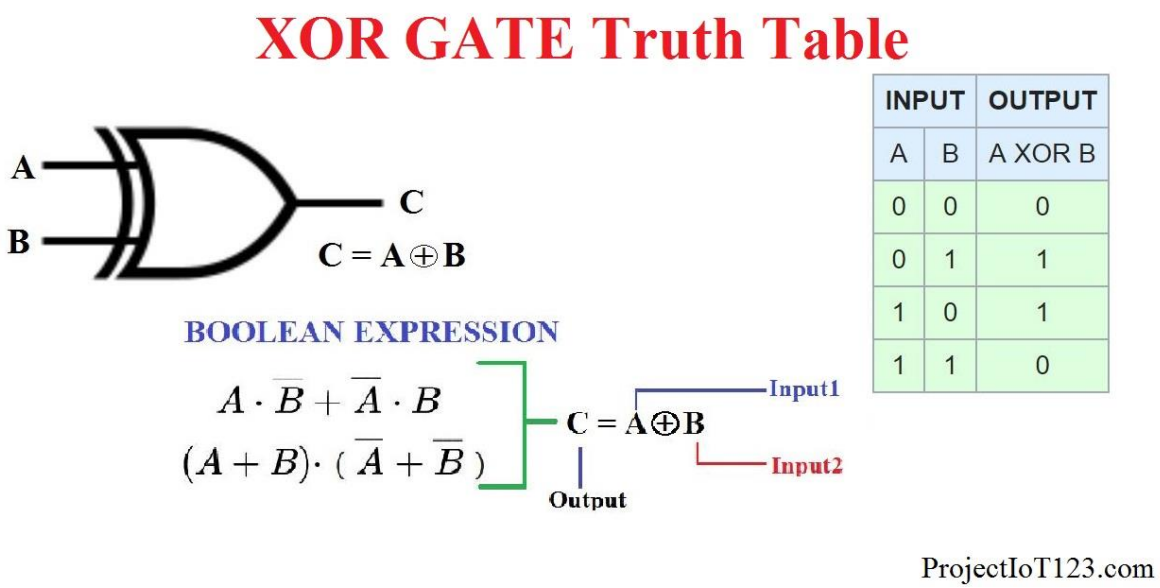
Vernam Cipher

Vernam Cipher Abstract

The Vernam cipher is, in theory, a perfect cipher. Instead of a single key, each plaintext character is encrypted using its own key. This key — or key stream — is randomly generated or is taken from a one-time pad, e.g. a page of a book. The key must be equal in length to the plaintext message. The fact that each character of the message is encrypted using a different key prevents any useful information being revealed through a frequency analysis of the ciphertext.

Concept in C/C++ programming

To encrypt the message, each character of the plaintext and the key will need to be converted to a numeric code. Fortunately, there are already coding schemes to do this, and we can use standard ASCII codes. As you may already know, in the ASCII coding system, each character is given a numeric code. For example, the letter 'H' is 72. This number has a binary representation of 01001000 (using 8 bits).



INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

***Figure 1** : XOR Truth Table*

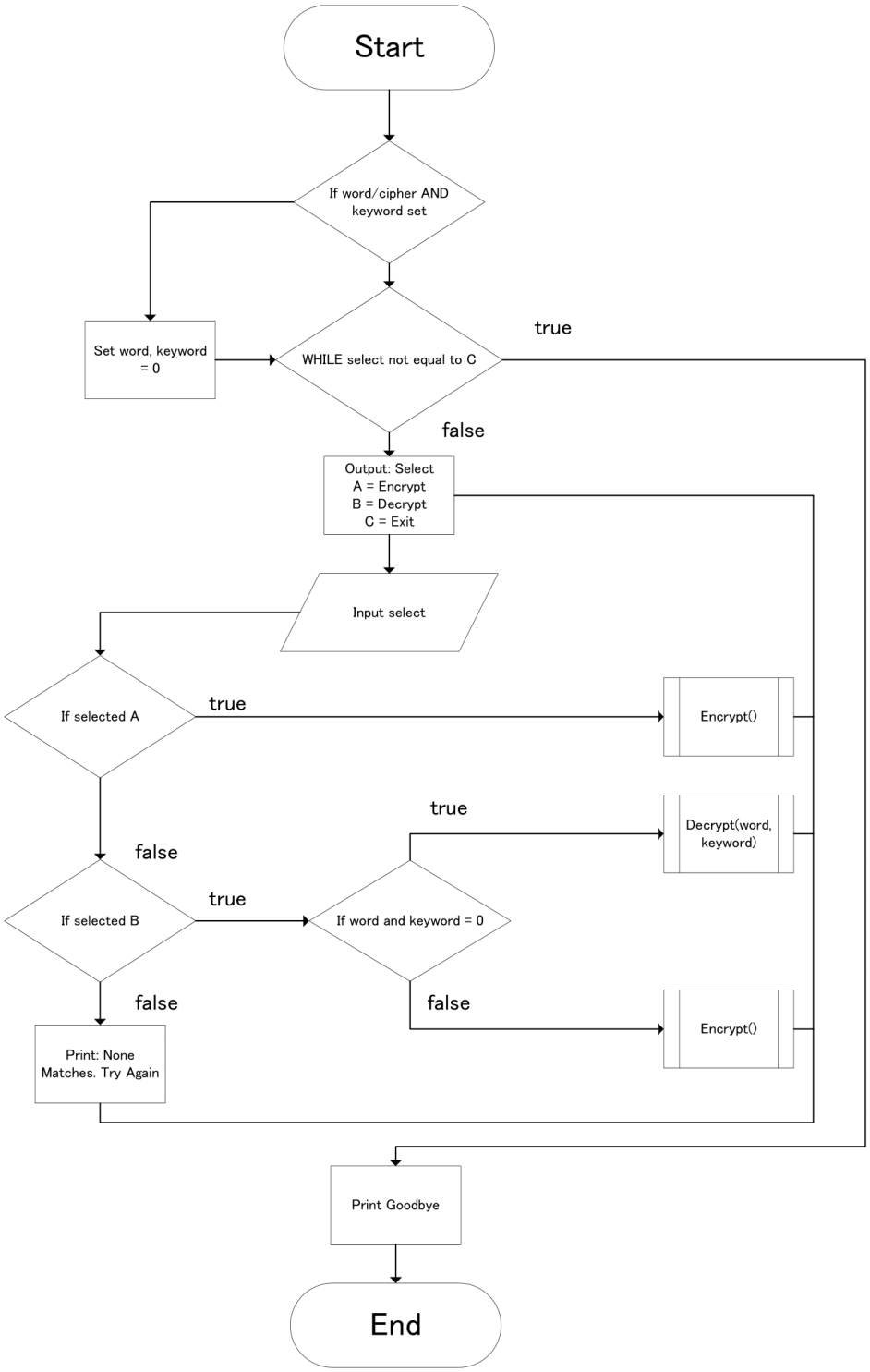
To apply the Vernam cipher, each bit of the binary character code for each letter of the plaintext is XOR'd with the corresponding bit of each letter of the binary character code for the corresponding character from the key stream — this creates the ciphertext.

Flowchart

The program that we coded in has three distinct functions that separate each unique sets of methods.

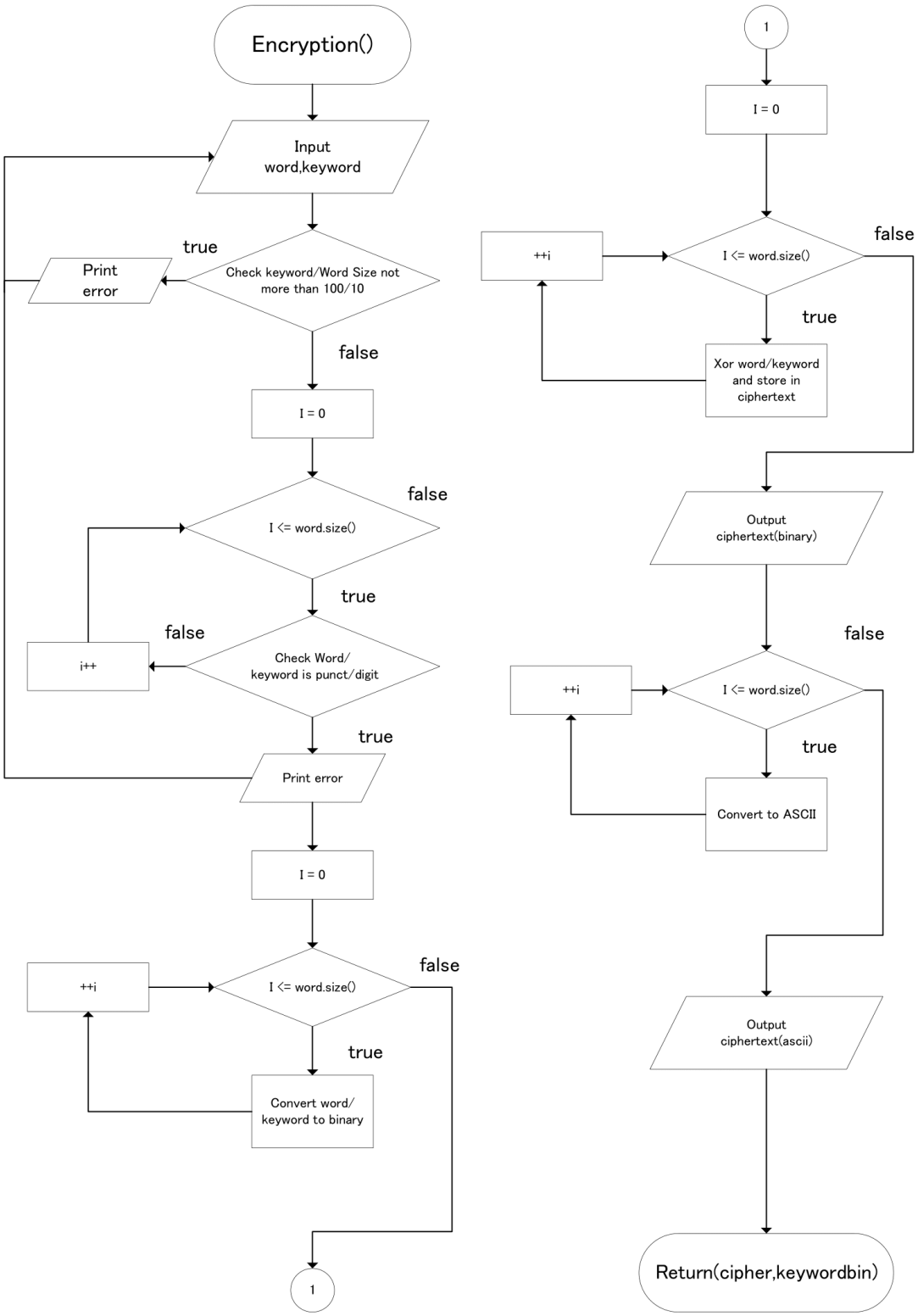
Flowchart 1 : Menu Function

Flowchart Function 1: Vernam Cipher Menu Function



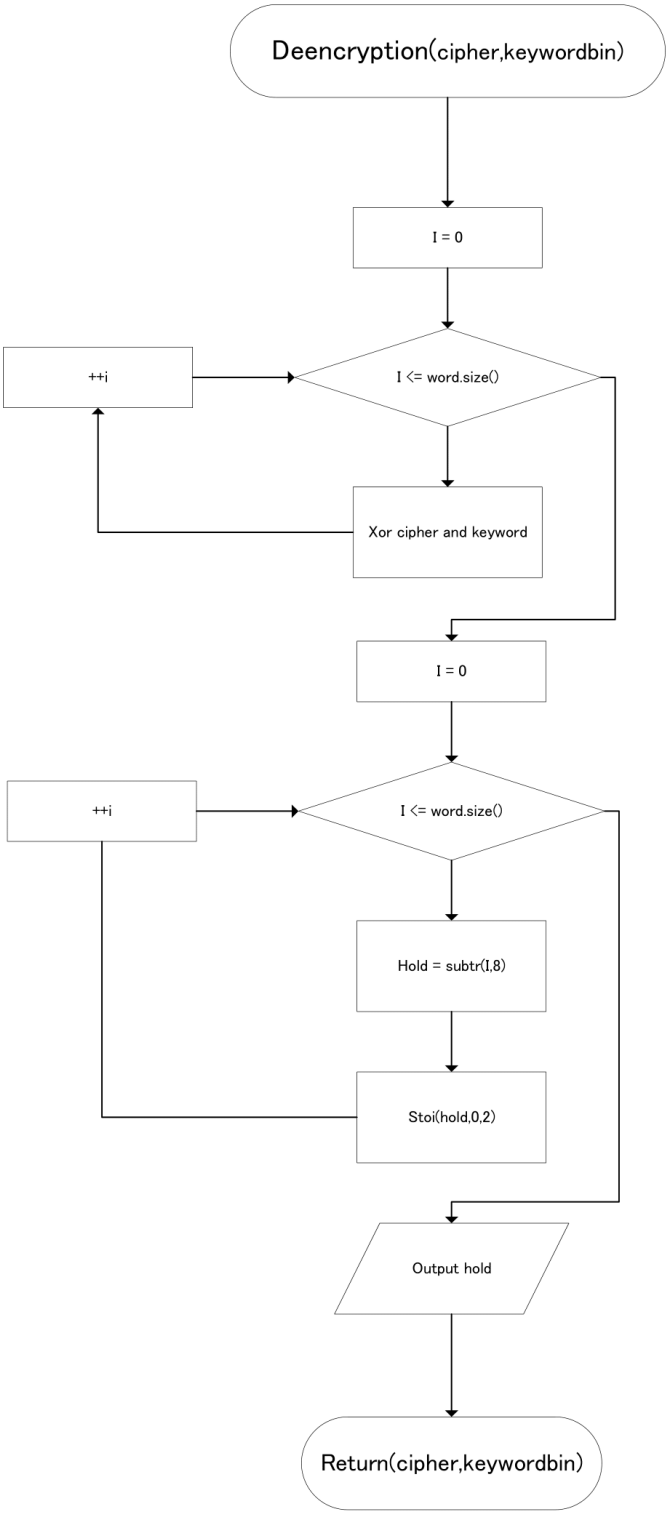
Flowchart 2 : Encryption Function

Flowchart Function 2: Vernam Cipher Encryption Function



Function 3 : Decryption Function

Flowchart Function 3: Vernam Cipher Decyrption Function



Coding

```
/*-----------------------------------------
Assignment I : Computer Security
Section 38

Vernam Cipher
Ahmad Faris Aiman bin Arizal
Ahmad Danial Harith bin Ahmad Kamal
Adam Ilman bin Suhaimi
-----------------------------------------*/

#include <iostream>
#include <stdlib.h>
#include <unistd.h>
#include <bitset>
using namespace std;


//function declaration
int menu(string ciphertext,string cipherkeywordtext);
int encryption();
int decryption(string ciphertext, string cipherkeywordtext);


int encryption(){
    //declaration
    string word,keyword;

    restart:
    system("CLS");
    cout << "*************************\n* Vernam Cipher Program *\n*************************";
    cout << "\n\nEncryption process\n\n";


    //enter word and keyword
    cout << "Enter the plaintext word: ";
    cin  >> word;
    cout << "\nEnter the keyword: ";
    cin  >> keyword;

    //check for size if similar or less
    //check if there is symbol or word in keyword/word
    if(keyword.size() > word.size() && keyword.size() >= 10 && keyword.size() <= 100 && word.size()){
        cout << "Program cannot continue " << endl
             << " Your word size: " << word.size() << endl
             << " Your keyword size: " << keyword.size() << endl;
        goto restart;
    } else {
        for(int i=0;i<word.size();i++){
            if(ispunct(word[i]) || ispunct(keyword[i]) || isdigit(word[i]) || isdigit(keyword[i])){ //check for symbols
                cout << "Program cannot continue. Your word/keyword contains symbols or numbers. " << endl
                     << " Your word contains: " << word << endl
                     << " Your keyword contains: " << keyword << endl;
                sleep(2);
                goto restart;
            }
        }
    }
    //encryption process

        //convert the plaintext to Binary
        string ciphertext,cipherkeywordtext;
        for(int i=0;i<word.size();++i){
            ciphertext.append(bitset<8>(word[i]).to_string());
            cipherkeywordtext.append(bitset<8>(keyword[i]).to_string());
        }
```

```
66
67          //fancy output
68          system("CLS");
69          cout << "*************************\n* Vernam Cipher Program
            *\n*************************";
70          cout << "\n\nEncryption process\n\n";
71          cout << "How it works" << endl << endl;
72
73          cout << "Before we proceed, we need to convert our word and keyword into binary
            " << endl
74               << "in order to use the XOR method of Vernam Cipher." << endl << endl;
75          sleep(2);
76
77          cout << "Word   : " << word << endl
78               << "Keyword: " << keyword << endl << endl;
79          sleep(2);
80
81          cout << "These word and keyword will be converted into binary as follows " <<
            endl << endl;
82          sleep(2);
83
84          cout << "Word    (in Binary form)    : " << ciphertext << endl
85               << "Keyword (in Binary form)    : " << cipherkeywordtext << endl;
86          sleep(2);
87
88          cout << "Understand the truth table XOR below and it will convert as follows :
            " << endl << endl
89               << "XOR truth table" << endl
90               << "A | B | Y" << endl
91               << "0 | 0 | 0" << endl
92               << "0 | 1 | 1" << endl
93               << "1 | 0 | 1" << endl
94               << "1 | 1 | 0" << endl << endl;
95          sleep(2);
96          cout << ciphertext << endl << cipherkeywordtext << endl;
97
98          //XOR operation
99          string result;
100         for(int i=0;i<ciphertext.size();++i){
101             if(ciphertext[i] == cipherkeywordtext[i])
102                 result += "0";
103             else
104                 result += "1";
105             cout << "-";
106         }
107
108     //output ciphertext
109     cout << endl << result << endl;
110
111
112     cout << endl << endl << "The encrypted text (ASCII) as follows: ";
113     //convert to binary to ascii
114     string placeholder;
115     for(int i=0;i<result.size();i=i+8){
116         placeholder = result.substr(i,8);
117         convert = stoi(placeholder, 0, 2);
118         ch = convert;
119         cout << ch;
120     }
121     cout << endl << "Encrypted Text (Binary XORed) : " << result << endl;
122     system("pause");
123
124     menu(ciphertext,cipherkeywordtext);
125 }
126
127 int decryption(string encrypted, string cipherkeywordtext){
128     //data declaration
129     int convert; //used in conjunction with char ch to convert from Dec to ASCII
130     char ch;
```

```
131
132         //fancy output
133         system("CLS");
134         cout << "*************************\n* Vernam Cipher Program
            *\n*************************";
135         cout << "\n\nDecryption process\n\n";
136         cout << "How it works" << endl << endl;
137
138         cout << "The user needs to have the ciphertext and the keyword obviously " << endl
139              << "in order to use the XOR method of Vernam Cipher." << endl << endl;
140         sleep(2);
141
142         cout << "Encrypted Text  (in Binary form)     : " << encrypted << endl
143              << "Keyword Text    (in Binary form)     : " << cipherkeywordtext << endl;
144         sleep(2);
145
146         cout << "These Cipher Text and Keyword Text will be XOR in order to get the
            original word. " << endl << endl;
147         sleep(2);
148
149         cout << "Understand the truth table XOR below and it will convert as follows : " <<
            endl << endl
150              << "XOR truth table" << endl
151              << "A | B | Y" << endl
152              << "0 | 0 | 0" << endl
153              << "0 | 1 | 1" << endl
154              << "1 | 0 | 1" << endl
155              << "1 | 1 | 0" << endl << endl;
156         sleep(2);
157         cout << encrypted << endl << cipherkeywordtext << endl;
158
159
160         //decryption
161         string result;
162         for(int i=0;i<encrypted.size();++i){
163             if(encrypted[i] == cipherkeywordtext[i])
164                 result += "0";
165             else
166                 result += "1";
167             cout << result[i];
168         }
169
170         cout << endl << endl << "The word as follows: ";
171         //convert to binary to ascii
172         string placeholder;
173         for(int i=0;i<encrypted.size();i=i+8){
174             placeholder = encrypted.substr(i,8);
175             convert = stoi(placeholder, 0, 2);
176             ch = convert;
177             cout << ch;
178         }
179
180         cout << endl;
181         system("pause");
182         menu(encrypted,cipherkeywordtext);
183
184     }
185
186     int menu(string ciphertext,string cipherkeywordtext){
187         //declaration
188         char choice;
189
190         //menu list
191         do{
192             system("CLS");
193             cout << "*************************\n* Vernam Cipher Program
                *\n*************************";
194             cout << "\n\nYour word    : " << ciphertext << endl
195                  << "Your Keyword : " << cipherkeywordtext << endl;
```

```
196
197         cout << "\n\nMenu choices: \n A. Encryption \n B. Decryption \n C. Exit \n\n";
198         cout << "Enter your choice: ";
199         cin  >> choice;
200
201         switch(choice){
202             case 'A':
203             case 'a':
204                 //encryption
205                 encryption();
206                 break;
207             case 'B':
208             case 'b':
209                 if(ciphertext == "0"){
210                     cout << "Your word and keyword is not set. Redirecting to
                        encryption process." << endl;
211                     sleep(2);
212                     encryption();
213                 }
214                 decryption(ciphertext,cipherkeywordtext);
215                 break;
216             case 'C':
217             case 'c':
218                 break;
219             default:
220                 cout << "Invalid choice." << endl;
221                 sleep(2);
222                 goto restart;
223                 break;
224         }
225     }while(choice != 'c' || choice != 'C');
226
227     return 0;
228 }
229
230 int main(){
231     menu("0","0");
232 }
```

Program Output

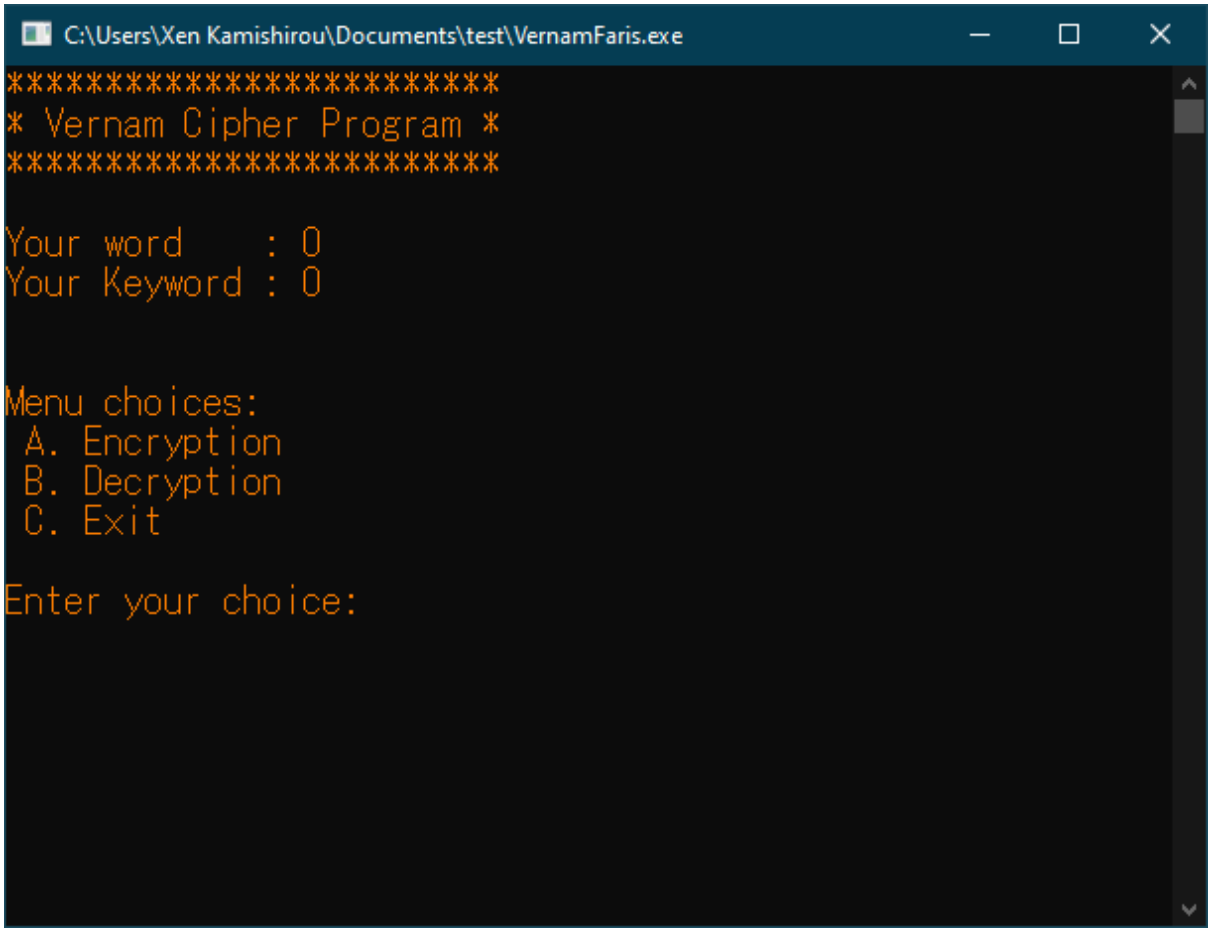


***Figure 2**: Main Menu*

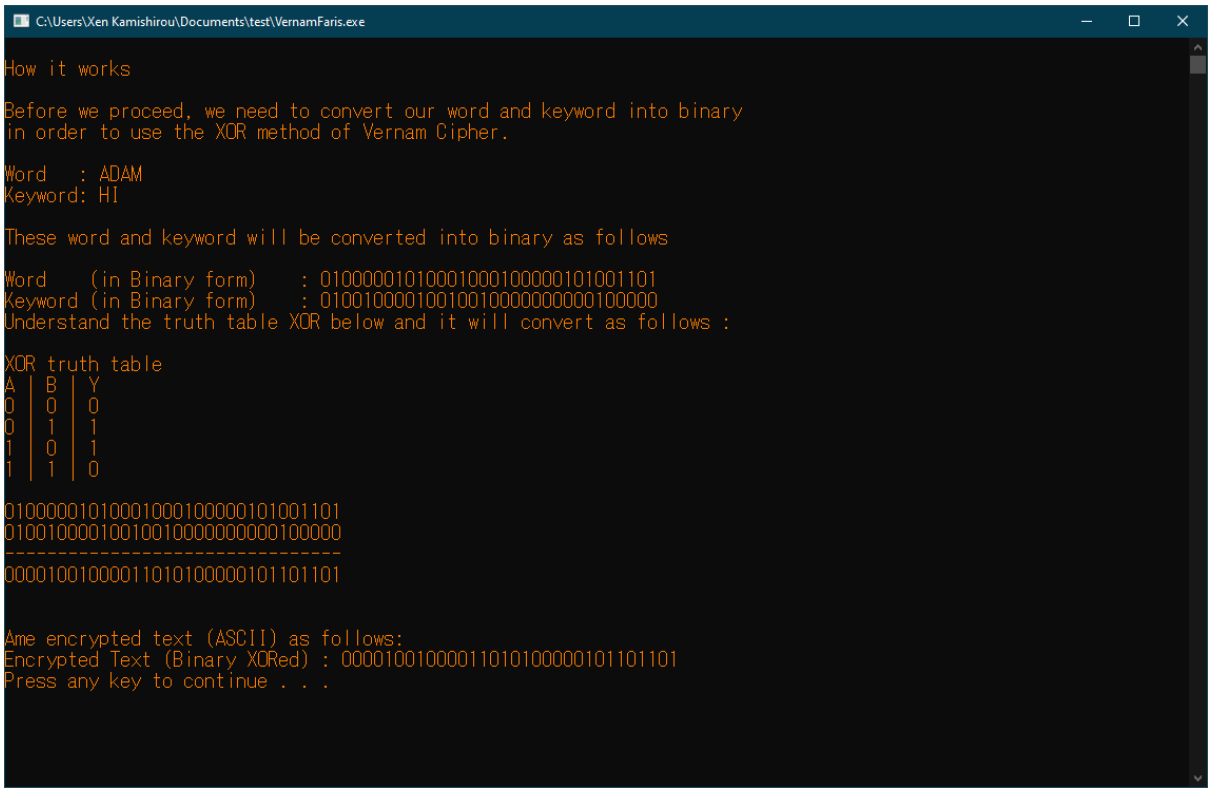


***Figure 3**: Encryption*

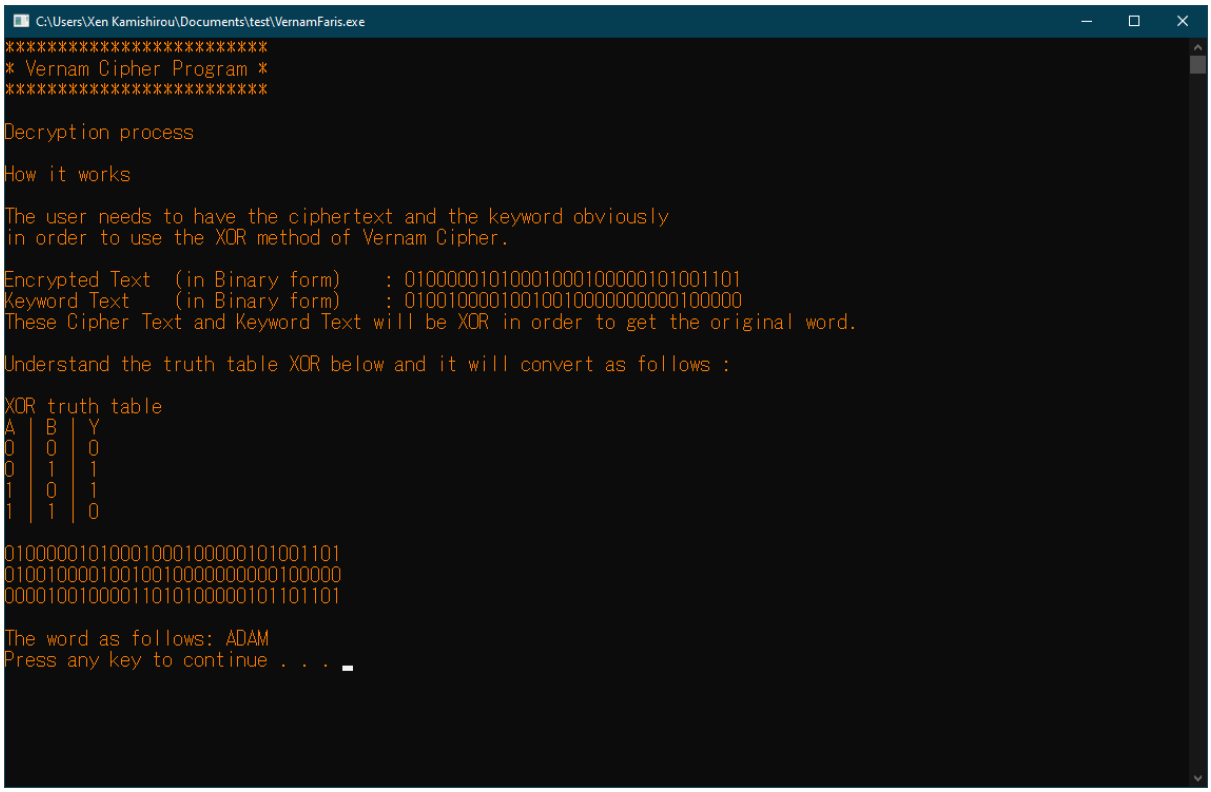


***Figure 4**: Decrypt*

Caesar Cipher

Caesar Cipher Abstract

Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

Concept in C/C++ programming

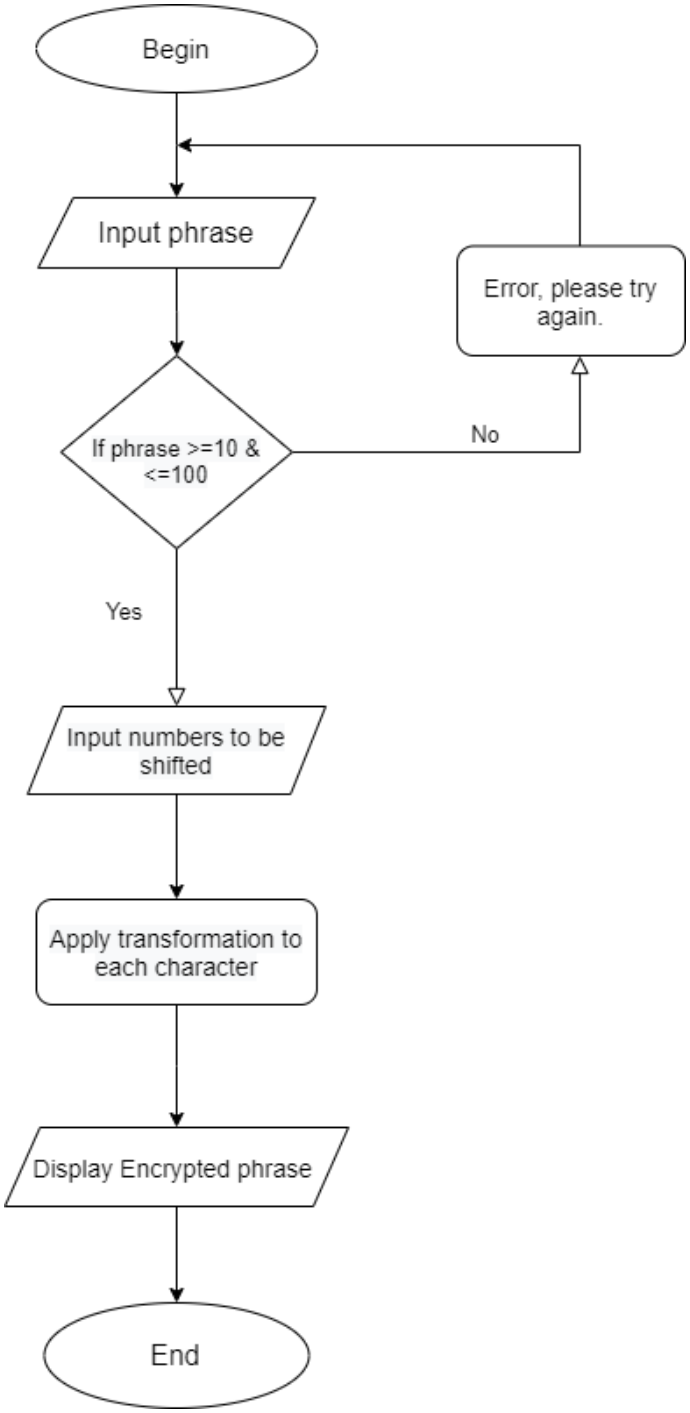
To encrypt message in Caesar Cipher, the user need to enter how many times do the program need to shift the message that have alphabetic, numeric, and special character in the program. Fortunately, there already coding schemes to do this, and we can use the standard dictionary which is The ASCII codes. As a general knowledge for computer science student, we already know that the ASCII code is numeric code that represent numeric, alphabetic, and special character in the table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

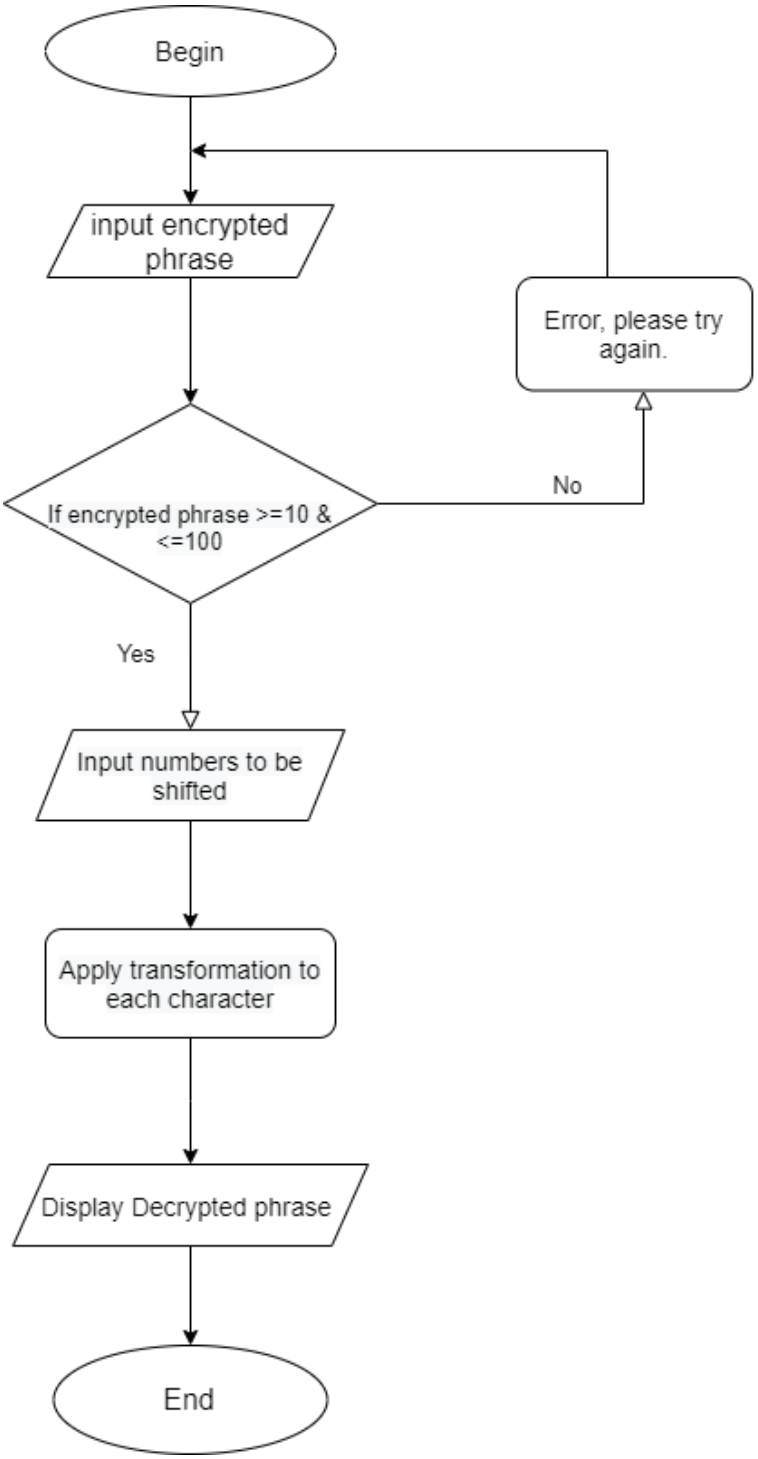
***Figure 5**: ASCII Table*

Flowchart

Ceaser Cipher Encryption



Ceaser Cipher Deencryption



Coding

```
/*
Assignment I : Computer Security
Caesar Cipher

Ahmad Faris Aiman bin Arizal
Ahmad Danial Harith bin Ahmad Kamal
Adam Ilman bin Suhaimi
*/

// A C++ program to illustrate Caesar Cipher Technique
#include <iostream>
#include <unistd.h>
using namespace std;

// This function receives text, shifts the text, and
// returns the encrypted text
string encrypt(string text, int s)
{
    string result = "";

    // Traverse text
    for (int i=0;i<text.length();i++)
        // Apply transformation to each character
        result += char(int(text[i]+s));

    // Return the result string
    return result;
}

string decrypt(string text, int s){

    string result = "";

    // Traverse text
    for (int i=0;i<text.length();i++)
        // Apply transformation to each character
        result += char(int(text[i]-s));

    // Return the result string
    return result;
}

// Driver program to test the above function
int main()
{
    int shift;
    string text,f,t;

    do{

        cout << "Enter a phrase to be encrypted.\n";
        getline(cin,text);
        cout << endl;

        if(text.size() >= 10 && text.size() <= 100){
            break;
        } else {
            cout << "error please try again" << endl;
            sleep(4);
            system("CLS");
        }

    }while(1);

    cout << "Enter how many letters to be shifted.\n";
    cin >> shift;

    f = encrypt(text, shift);

```

```
70        cout << "\nText         : " << text;
71        cout << "\nShift         : " << shift;
72        cout << "\nEncrypted     : " << f;
73
74        t = decrypt(f,shift);
75        cout << "\nDecryption : " << t;
76        return 0;
77    }
```

Program Output

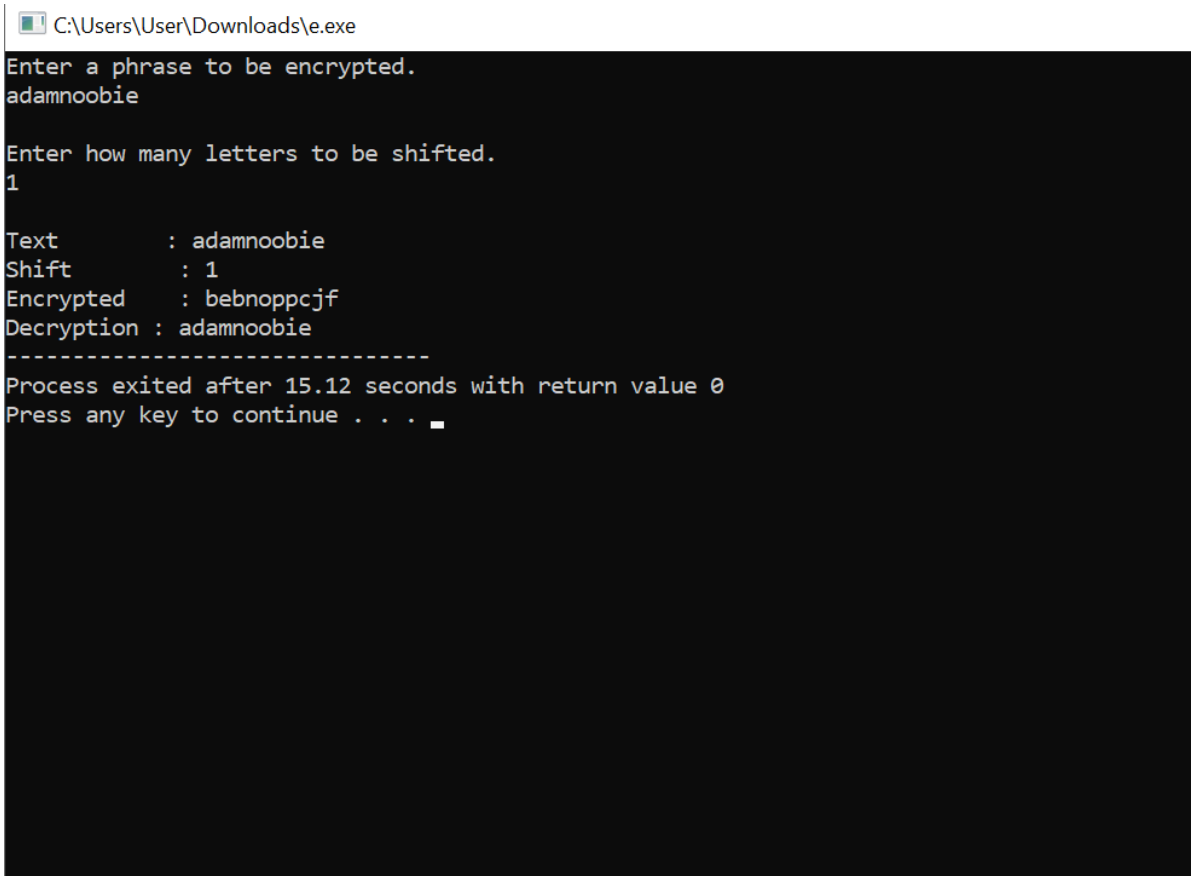


***Figure 6**: Encrypt & Decrypt*

References

Caesar cipher. (2020). Retrieved 26 July 2020, from https://en.wikipedia.org/wiki/Caesar_cipher

Flowchart Maker & Online Diagram Software. (2020). Retrieved 26 July 2020, from https://app.diagrams.net/