Encryption and Decryption using RSA

Group Members:

- Muhammad Zunique (20K-0145)
- Ammar Siddiqui (20K-0177)
- Muhammad Muaz Zaffar (20K-0363)

Computer Organization and Assembly Language

Course Code: EE-2003

National University of Computer and Emerging Sciences – FAST

INTRODUCTION:

Long ago, when some person wants to deliver the message from one place to another by making sure that the message should not leak the information. So, for that different Encryption and Decryption algorithms were being introduced amongst which RSA is considered as one of most prominent as it is wholly based on prime numbers and till yet in today's world no generalize formula has been invented that can decode prime numbers logic. And this logic of prime and large arithmetic values makes it more beneficial for encryption and decryption. There public and private keys that are used for encryption and decryption purpose respectively maintaining Asymmetric property of keys and modulo function including Euclidean.

Literature Review:

The background of this topic in computer science generally referred as CRYPTOGRAPHY as network security is pretty much necessary via developing it in today's modern era where hacking other's databases private information is just quiet easy, so one have to make sure that his security system must be equally strong via developing team. So these Encryption techniques provides the general idea that how on large network and datasets we can perform operation to secure our data. Mathematical calculations is a major part for doing RSA technique as we encode message to number and do manipulation on those numbers and the decode those using defined methodology

Problem Statement:

You must have seen when you open a chat on Whatsapp, a message shows at the top that chats are end to end encrypted what's that? That is what basically Encryption and Decryption is all about, securing messages from one end to another end. The main purpose is that the idea of message should not be leaked so for that different techniques are used amongst which we chose RSA as it is one of prominent

and not that much easy to decode as if consider some commonly used such as Ceaser cipher, Shift Cipher or Hill cipher as they do not require much effort to be decrypted.

Methodology(Tools and Techniques):

- Microsoft Visual Studio 2019
- IRVINE MASM library
- Relevant methods explained in code for calculating values for ciphering and Deciphering.
- Plain text to encrypted text message
- Encrypted Text to plain/ Decrypted Text message

Detailed Design and architecture:

First take input two prime numbers by means all which rest of values are dependent. Then the public and private keys are generated on which encoding and decoding is dependent.
PUBLIC KEY : (e,n) // n = p*q and k = (p-1)(q-1) and 1 < e < k
PRIVATE KEY : (d,n) // where is inverse of e

The calculations of following components are defined in code explicitly.

$$C = m^e \bmod(n) \qquad m = C^d \bmod(n)$$

These are two formulae used for encryption and Decryption.

Implementation, Testing and Code:

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc
.data
	;//--Two  prime Numbers--//
	p DWORD ?
	q DWORD ?
	;//----------------------//

	n DWORD ?
	k DWORD ?
	temp2 DWORD 1

	;//---Encrption and decryption key--//
	e DWORD ?  ; is the encryption key
	d DWORD ?  ; is the decryption key
	;//----------------------//

	temp DWORD ?
	save_eax DWORD ?
	buffer DWORD 500
	message BYTE 500 DUP(?)
	char_entered DWORD ?

	temp_array SBYTE 500 DUP(?)
	encrypted BYTE 500 DUP(?)
	decrypted BYTE lengthof encrypted DUP(?)

	is_prime PROTO, input:DWORD


;//-------------------------MAIN PROGRAM--------------------------//
.code
	main PROC

		;//----clearing resgiters----//
		xor eax,eax
		xor ebx,ebx
		xor ecx,ecx
		xor edx,edx
		;//--------------------------//

		;//-----------Formatting of code Deisgn-----------//
		menu:
```

```
                mWrite"                          ENCRYPTION AND
DECRYPTION USING RSA ALGORITHM "
                call crlf
                mWrite"
========================================== "
                call crlf
                call crlf
                call crlf


           ;//----Taking INPUT----//
           call take_input
           mov eax,p
           cmp eax,q
           je invalid
           ;//--------------------//

           ;//----check for isPrime----//
           invoke is_prime,p
           cmp eax,1
           je Invalid
           invoke is_prime,q
           cmp eax,1
           je Invalid
           ;//-------------------------//

           ;//----if isprime successful calculate follweoing-----//
           call gen_n
           mov n,eax
           call gen_k
           mov k,eax
           sub eax,1
           mov e,eax
           ;//-----------------------------------------------------//

           ;//-------Three major Functions------//
           call show_values
           call encrypt
           call decrypt
           ;//-----------------------------------//

           jmp endd

           ;//---------Labels--------//
           Invalid:
                mwrite "Invalid input "
```

```
          endd:
               exit
          ;//----------------------//
     main ENDP


;//---Check for whether the numbers entered are prime or not-----//
is_prime PROC , input:DWORD
     mov ebx,2
     check_loop:
          xor edx,edx
          mov eax,input
          div ebx
          cmp edx,0
          je n_prime
          inc ebx
          cmp ebx,input
          jne check_loop

     mov eax,0       ; 0 represents is prime
     ret
     n_prime:
          mov eax,1       ;1 represents not prime
     ret
is_prime ENDP

;//-----------------------------------------------------------------------
---//


;//---------Display the calculated values------------//
show_values PROC
     call crlf
     call crlf
     mWrite"Calculated values are : "
     call crlf
     mWrite"========================"
     call crlf
     call crlf
     mov eax,p
     mwrite "The value of first prime number is: "
     call writedec
     call crlf
     mov eax,q
     mwrite "The value of second prime number is: "
     call writedec
```

```
     call crlf
     mwrite "The value of n : "
     mov eax,n
     call writedec
     call crlf
     mwrite "The value of k : "
     mov eax,k
     call writedec
     call crlf
     mwrite "The value of e : "
     mov eax,e
     call writedec
     call crlf
     call crlf

     ret
show_values ENDP
;//-------------------------------------------------//


;//-----------converting plain text to encrypted text-----------//
encrypt PROC


     xor edx,edx
     xor esi,esi
     xor edi,edi

     ;//---for calculating length of string so that loop can be calculates---//
     mov ecx,char_entered

;//---Loop1 is running ascii equivalent codes of every character of string----//
     l1:
           mov bl,message[esi]
           movzx ebx,bl
           sub ebx,60h    ;//--subtracting 60 as per logic--//
           mov eax,1
           mov temp,ecx
           mov ecx,e

          ;//---in LOOP2 multiplying it as per formula---//
          l2:
                imul eax,ebx
                xor edx,edx
                mov save_eax,eax
                cdq
```

```
                idiv n   ;//---for getting mod calculation--//
                mov eax,save_eax
                mov eax,edx
          LOOP l2

          mov temp_array[esi],al
          add eax,60h    ;/adding that 60 again that we first subtracted so that correct encryptes string is obtained
          mov encrypted[edi],al
          inc edi
          inc esi
          mov ecx,temp
          ;//---clearing registers---//
          xor eax,eax
          xor ebx,ebx
     LOOP l1

     ;//----show the enrypted string----//

     mWrite"--------------------ENCRYPTED CONTEXT-------------------"
     call crlf
     call crlf
     mov esi,offset encrypted
     mov ecx,char_entered
     mWrite"Encrypted Key values are : "
     l4:
          mov al,[esi]
          call writeInt
          mWrite" "
          inc esi
     LOOP l4

     call crlf
     call crlf
     mov esi,offset encrypted
     mov ecx,char_entered
     mwrite "The encrypted data is : "
     l3:
          mov al,[esi]
          call writeChar
          mWrite" "
          inc esi
     loop l3
     call crlf
     call crlf
     ret
encrypt ENDP
```

```
;//-----------------------------------------------------------------//


;//---------Take Input prime Numbers--------------//
take_input PROC

     mwrite"Enter any two distinct prime numbers(whose product is less than 128): "
     call readInt
     mov p,eax
     call readInt
     mov q,eax
     call crlf
     mwrite "Enter the message want to encrypt : "
     mov ecx,buffer
     mov edx,offset message
     call readString
     mov char_entered,eax

     ret
take_input ENDP
;//-------------------------------------------------//


;//---------- Generate k ----------------------//
gen_k PROC
     mov eax,p
     mov ebx,q
     sub eax,1
     sub ebx,1
     imul eax,ebx
     ret

gen_k ENDP
;//-----------------------------------------//


;//---------- Generate d ----------------------//
gen_d PROC
     xor eax , eax
     xor ebx , ebx
     mov edx , 0

     Lb1:
          mov eax , n   ;moving value of  n in eax
          mul temp2     ; multiplying it with  a variable defined above and taking floor value
          add eax , 1
```

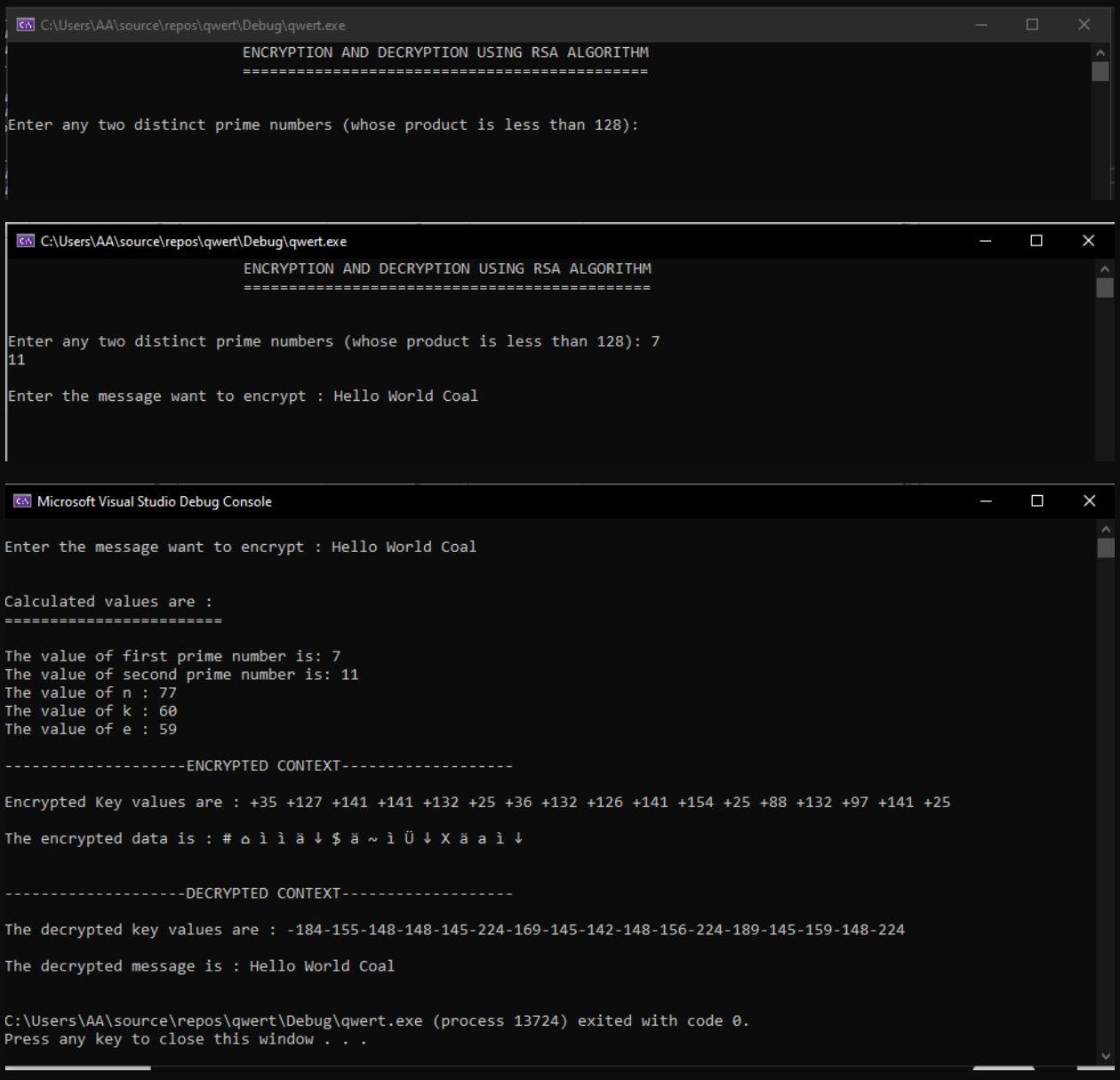
```
          mov ebx , e
          div ebx       ; for getting eax -> qoutient & edx ->
remainder
          cmp edx ,0 ;if remainder zero jump to label and set that paricular value present in eax to 'd'
          jmp setValue
          add temp2 , 1  ; else loop until zero
          JMP Lb1

     setValue:
          mov d , eax

     ret
gen_d ENDP
;//-----------------------------------------//


;//---------- Generate n ----------------------//
gen_n PROC
     mov eax,p
     mov ebx,q
     imul eax,ebx
     ret
gen_n ENDP
;//-----------------------------------------//

;//-------------------Decryption Function-----------------------//
decrypt PROC

     ;//---setting values---//
     mov esi,offset temp_array
     mov edi,offset decrypted
     mov ecx,char_entered


     outerloop:
          ;//--clearing registers--//
          xor eax,eax
          xor ebx,ebx

          mov al,[esi]
          movsx eax,al
          mov ebx,1
          mov temp,ecx
          mov ecx,e
```

```
     innerloop:
          xor edx,edx
          imul ebx,eax          ;k = ebx
          mov save_eax,eax
          mov eax,ebx
          cdq
          idiv n    ;quotient gets in eax and remainder in edx
          mov ebx,edx
          mov eax,save_eax
     LOOP innerloop
     mov eax,ebx
     add ebx,96
     mov [edi],bl
     mov ecx,temp
     inc esi
     inc edi
LOOP outerloop
call crlf

;//---Displaying Decrypted Text----//

mWrite"--------------------DECRYPTED CONTEXT-------------------"
call crlf
call crlf
mov esi,offset decrypted
mov ecx,char_entered
mwrite "The decrypted key values are : "
L1:
     mov al,[esi]
     call writeInt
     inc esi
loop L1
call crlf
call crlf

mwrite "The decrypted message is : "
mov esi,offset decrypted
mov ecx,char_entered
L5:
     mov al,[esi]
     call writechar
     inc esi
loop L5
call crlf
call crlf
ret
```

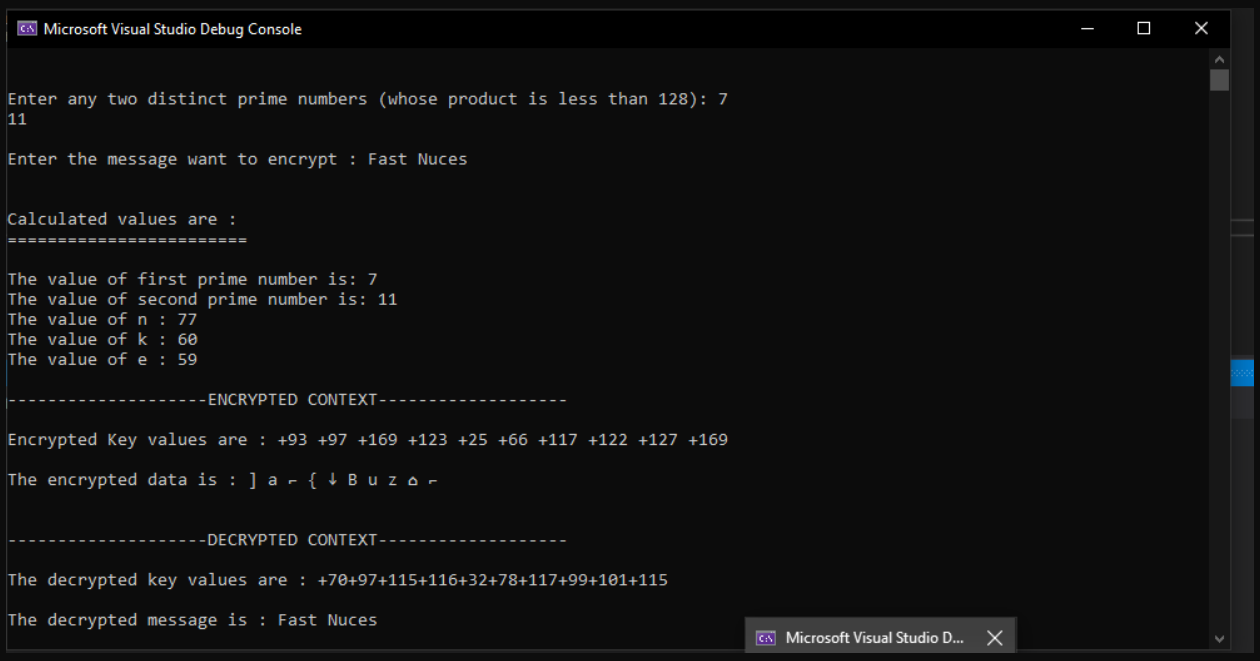
```
decrypt ENDP
;//-------------------------------------------------------------//

END main

;//----------------------MAIN ENDED-----------------//
```

Results:

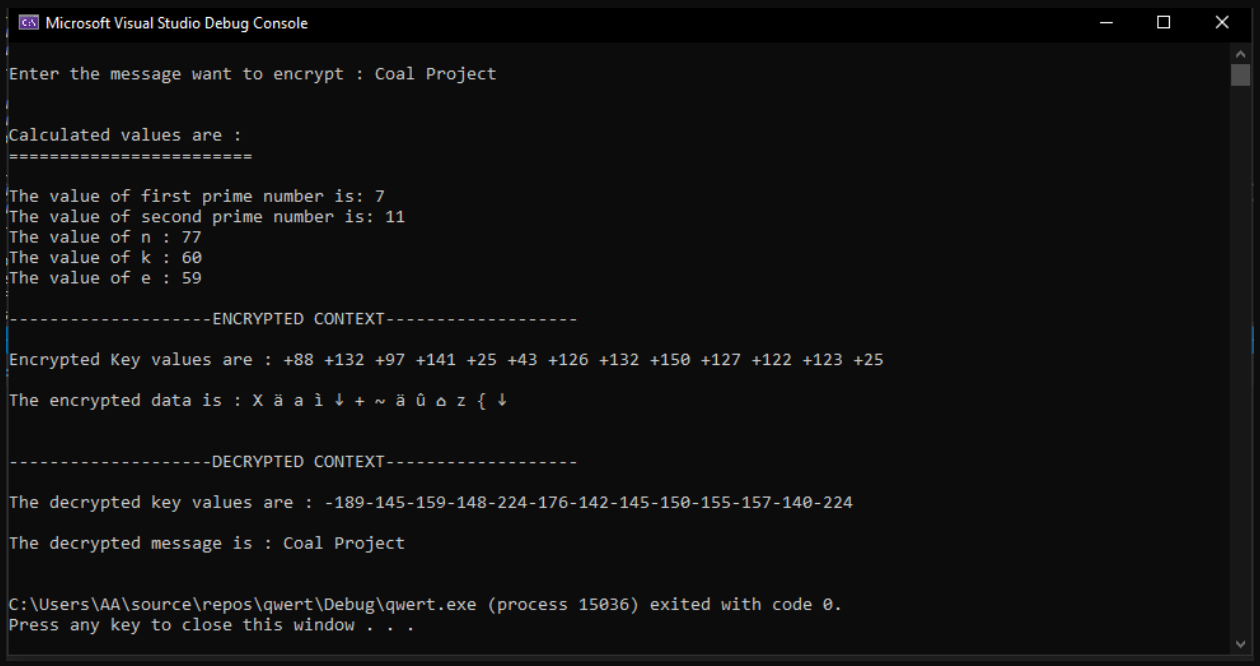
SAMPLE 1:



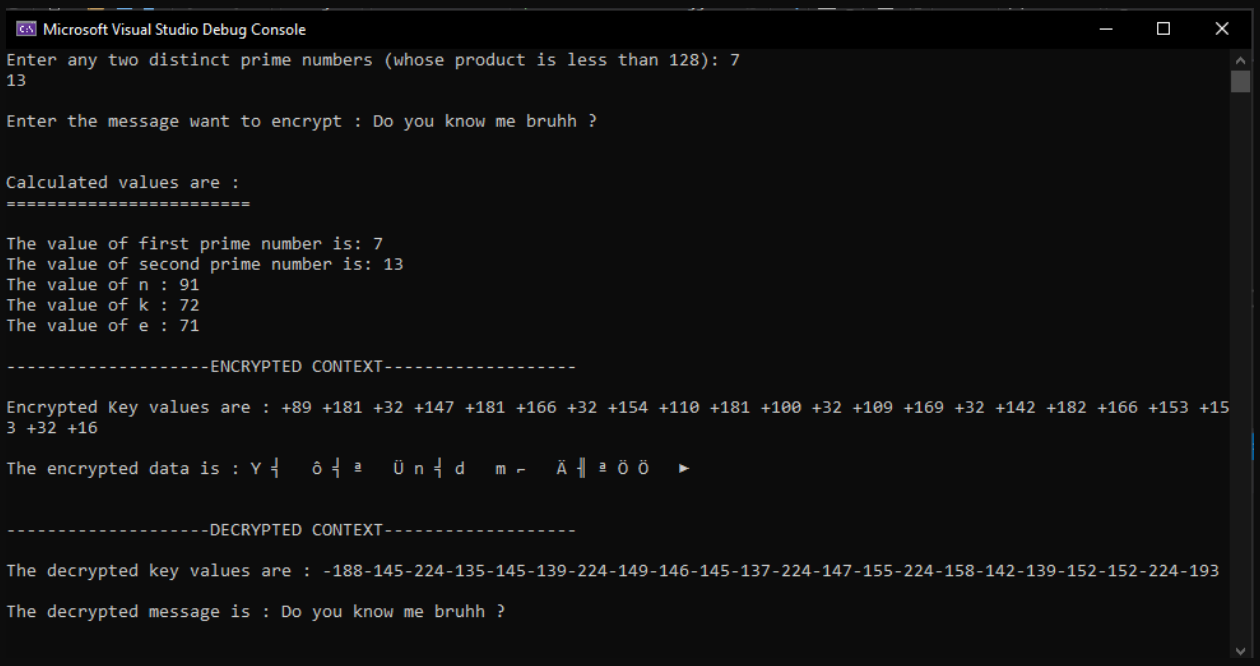
SAMPLE 2:



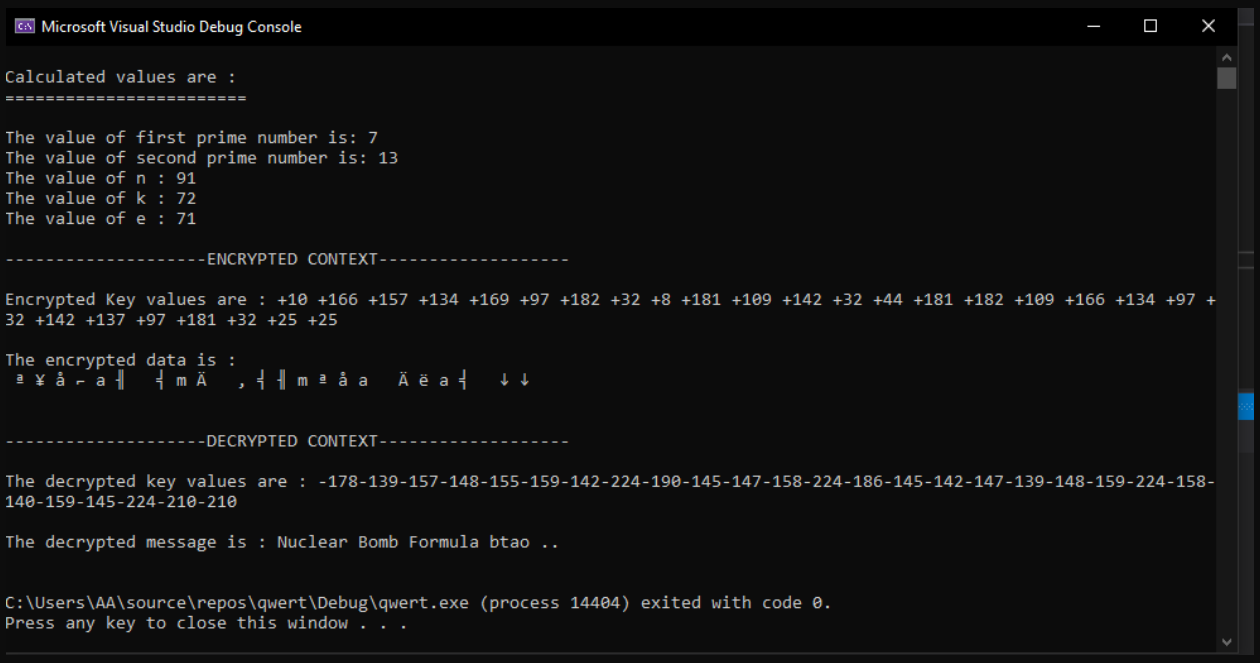
SAMPLE 3:



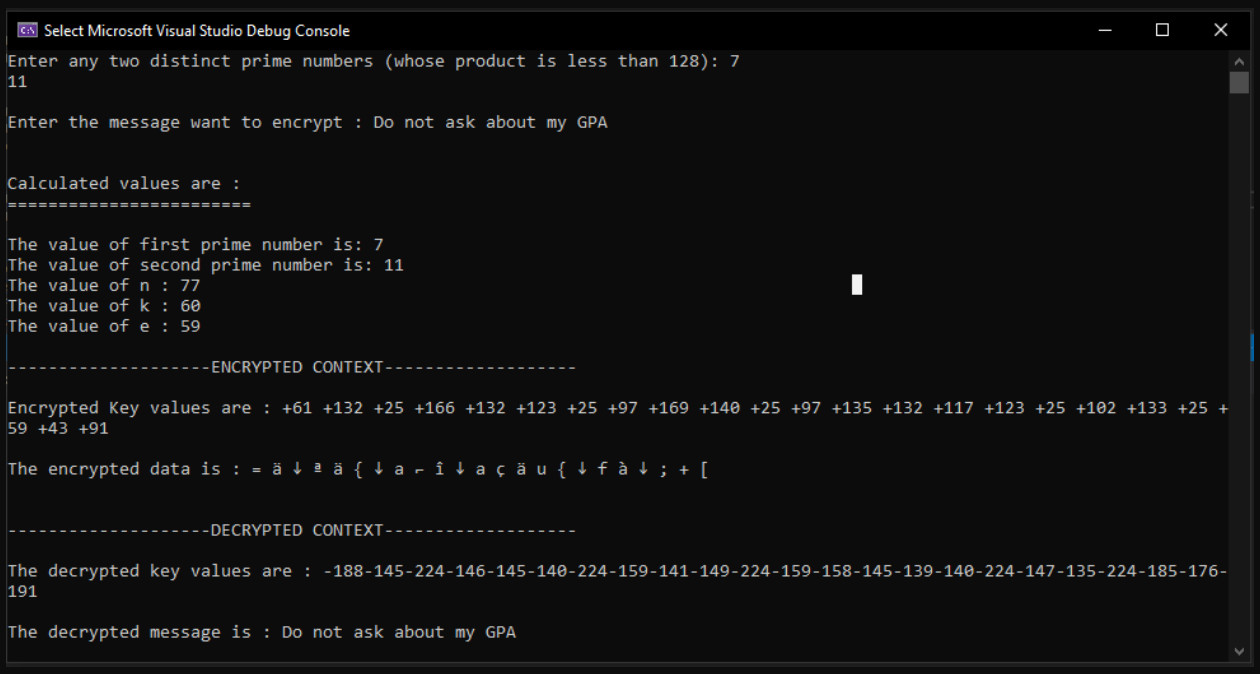
SAMPLE 4:



SAMPLE 5:



SAMPLE 6:



Conclusion:

Our main goal in the statement was that we should secure the message from being its get stolen by means of ciphering and provide the key to only the person we want to give or provide the formula of decryption. So that the message should remain end to end encrypted and cipher everyone can see / read but unable to understand the meaning of it until properly decoded into plain text.