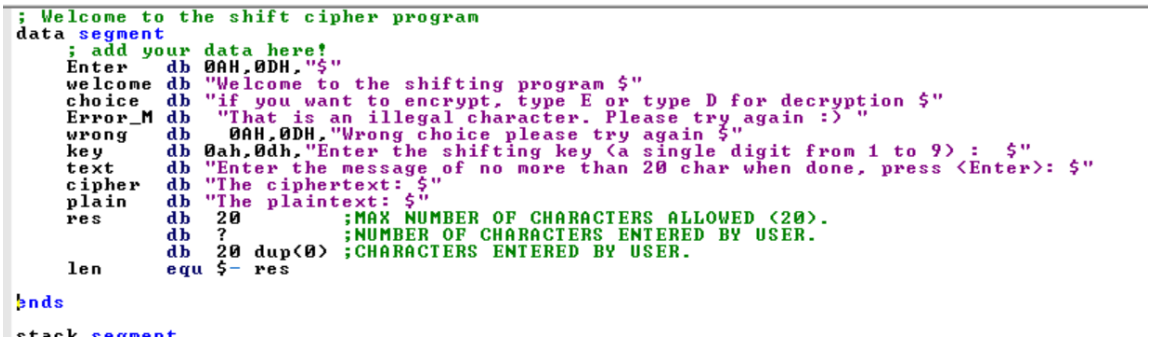
Name :
Leen Abdallah 20200512
Noor Taher 20200619
Jenin Al-Nayrab 20200626

Assembly Project (1)
Shifting cipher

This program shows the algorithm of shifting cipher by entering the plaintext and getting the ciphertext as letters of the original text but shifted depending on fixed number which is the key entered by the user from 0-9 (by assuming it's an integer of only one digit)

Firstly, the code segment is the place where we declare the variables and pre-define the output statements so the user would be able to understand the thing he/she would enter



```
; Welcome to the shift cipher program
data segment
    ; add your data here!
    Enter    db 0AH,0DH,"$"
    welcome  db "Welcome to the shifting program $"
    choice   db "if you want to encrypt, type E or type D for decryption $"
    Error_M  db  "That is an illegal character. Please try again :) "
    wrong    db   0AH,0DH,"Wrong choice please try again $"
    key      db 0ah,0dh,"Enter the shifting key (a single digit from 1 to 9) :  $"
    text     db "Enter the message of no more than 20 char when done, press <Enter>: $"
    cipher   db "The ciphertext: $"
    plain    db "The plaintext: $"
    res      db  20          ;MAX NUMBER OF CHARACTERS ALLOWED (20).
             db  ?           ;NUMBER OF CHARACTERS ENTERED BY USER.
             db  20 dup(0) ;CHARACTERS ENTERED BY USER.
    len      equ $- res

ends

stack segment
```

Enter : will be used in NewL function to only calle it if we want to add NewLine
Welcome: is the message that will be displayed at the start of the program to tell the user that this program will create a shifting algorithm
Choice : for asking for the choice that will be stored in [3100H]
Error_message (which is an extra) that will Display error message if we enter neight D nor E and ask again for entering an appropriate choice
Key: will ask for the key entering to use it in the shifting
Text: we will display an entering message to enter the plaintext or the ciphertext
Cipher → will be displayed for the E as message with the result
Plain → will be displayed for the D as message with the result
Res: for storing the message which is max to 20 char
len for the looping (but it should be used for array that is already declared in the data segment)
And by this the code segment ends

```
code segment

globalization:
                mov ax, data
                mov ds, ax
                mov es, ax
                ret
program_info:
                ;only for displaying the output, input and string so that the user can understand
                call NewL
                mov ah, 9
                mov dx, offset welcome
                int 21h
                call NewL
                ret
choicee:
                mov ah, 9
                mov dx, offset choice
                int 21h
                mov AH,1
                int 21h
                cmp al,'E'
                JE Read ; if staetment
                cmp al,'D'
                JE Read ; if else
                call NewL
                mov ah, 9
                mov dx, offset Error_M
                int 21h
                call NewL
                mov ah, 9
                mov dx, offset Wrong
                int 21h
                JMP choicee;to enter a valid choice which is the else staetment
                ret
```

For the code to be redable i separated the code into separated functions (that well be called from the start: in the rest of the code) .Firstl, the function globalization by getting the address of the data segment , the program info would only display to the screen the welcoming message <Welcome to the shift program> using the software interrupts 21h by using service #9
Lastly choices will also display the message <enter E for encrypt, D for decrypt>
And get the choice from the user that will be stored in al
And after this compare al (where the choice is stored) with ‘D’ or ‘E’ if one is satisfied the assembler will jump to read function otherwise it will enter Error message that <Illegal input> and asking the user to enter for valid input such as E or D

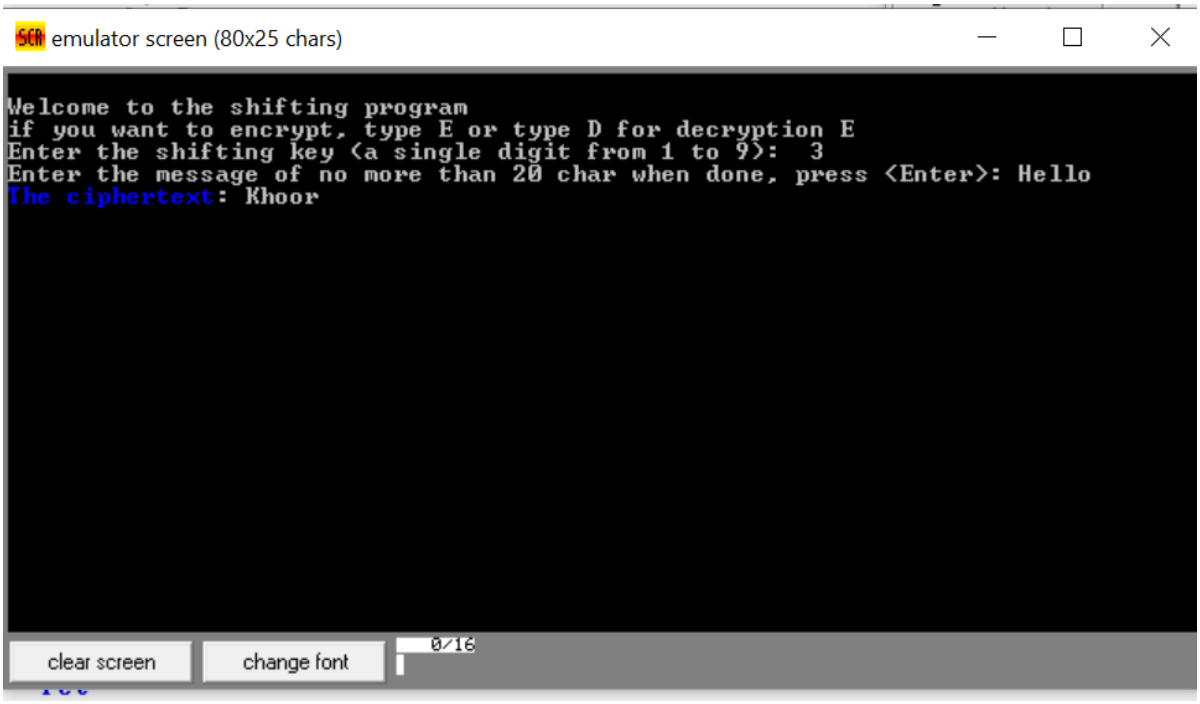
```
Read:
                    mov [3100h],al ; save the choice as the qustion wants
                    mov ah, 9
                    mov dx, offset key
                    int 21h
                    mov ah,1
                    int 21h
                    sub al,30h
                    mov [3101h],al
                    cmp [3100h], 'D'
                    JE Decode
                    call NewL
                    mov ah,9
                    mov dx, offset text
                    int 21h
                    mov ah, 0Ah ;SERVICE TO CAPTURE STRING FROM KEYBOARD.
                    mov dx, offset res
                    mov [3102h],dx
                    int 21h
                    call NewL
                    call encrypt
                    ret
```

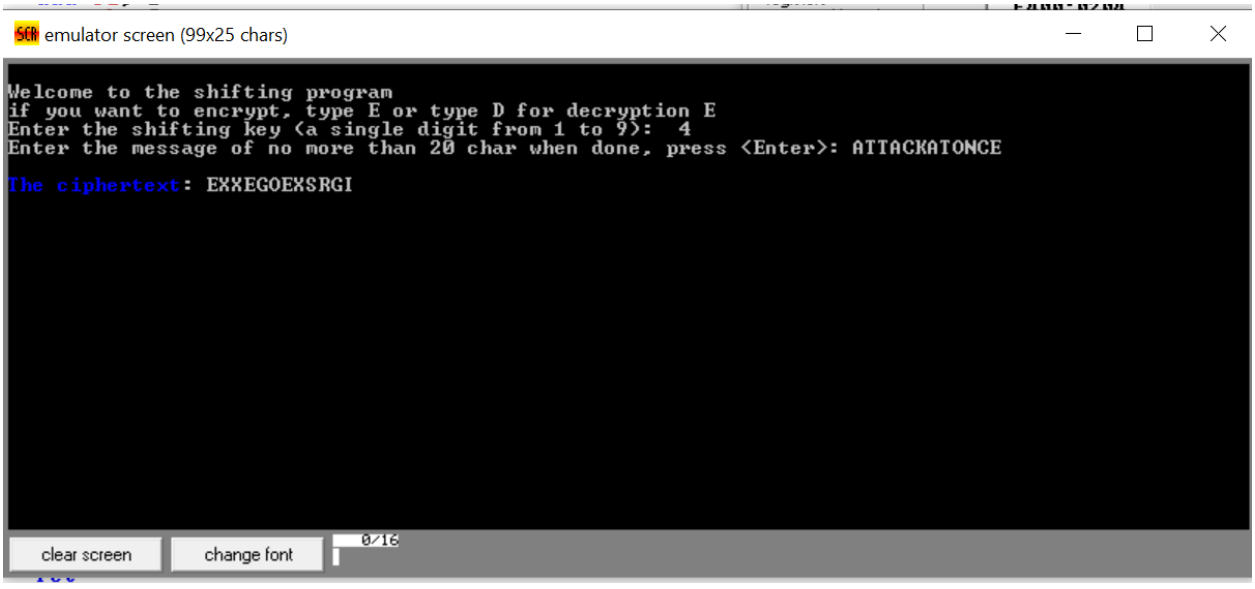
The read function stores from al which is the choice in [3100H] then enter the message that will ask for the key value and then enter it by the user to be stored in [3101H] then the choice will be compared with the stored choice in [3100H] if it was ‘D’ then decrypt otherwise enter the message and call encrypt function

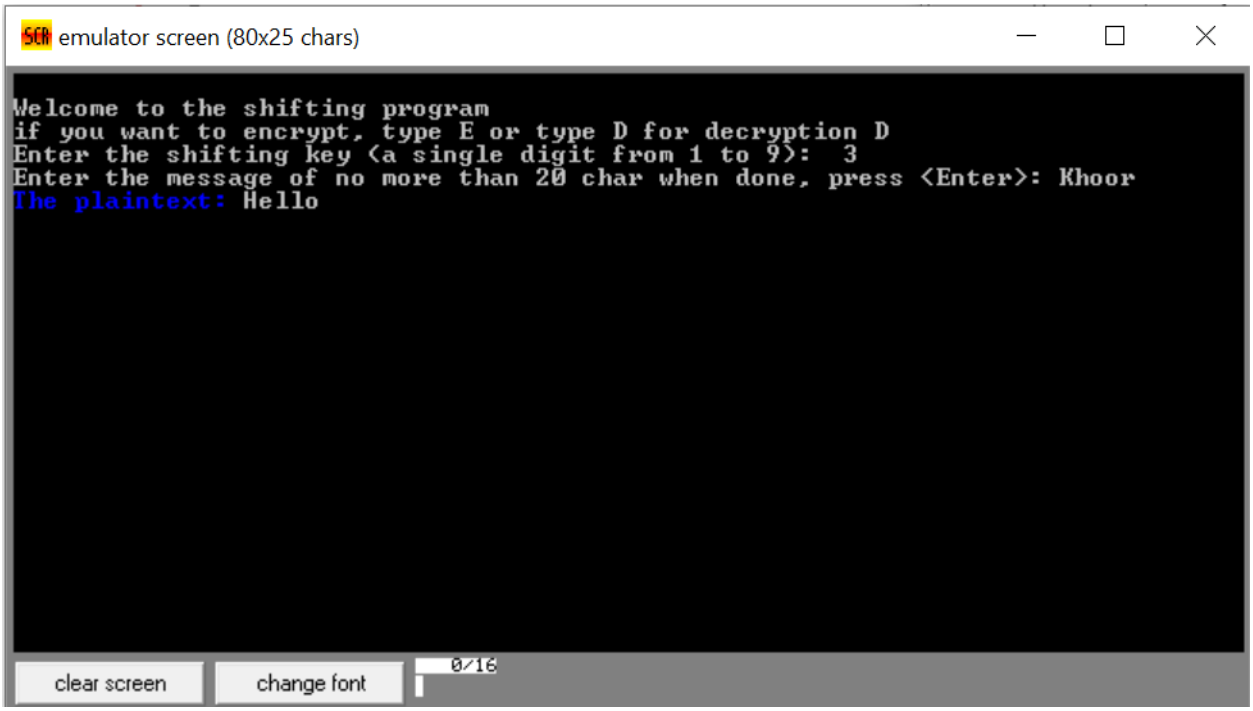
```
Encrypt:
                mov si, offset res
                add si, 1
                mov ch, 0
                mov cl,[si]
                ;push cx

                call Repeat
                ret       ; we will use it in the decryption
Repeat:
                inc si
                mov al,[si]           ;GET THE CHAR FROM BUFFER
                add al,[3101H]        ;ADD KEY
                mov [si-2],al         ;STORE IT TO THE BEGINING OF THE BUFFER
                loop Repeat            ;REPEAT
                mov [si],'$'          ;ADD END OF STRING TO THE LAST 2 LOCATIONS
                mov [si-1],'$'
                mov ah, 09h
                mov ah,9
                mov ah,09h
                mov bl,9
                mov cx, 14  ; mov to cx number of char
                int 10h
                mov dx,offset cipher
                int 21h
                mov ah,9
                mov dx, offset res    ;DISPLAY THE STRING AFTER ENCRYPTION
                int 21h
                ret
```

The encryption and the decryption function are the same but the formula differs by encryption by adding the key otherwise by subtracting the key from the ascii code of each letter
First the offset of the array will be stored in si to access the elements (the loop will still looping depending on the ascii code of the second element that will work on 32 character string (we won't encrypt or decrypt more than 20 char string))
And at the end the main function will be called from the start of the code segment and other function we will call them from other functions


emulator screen (80x25 chars)
Welcome to the shifting program
if you want to encrypt, type E or type D for decryption E
Enter the shifting key (a single digit from 1 to 9): 3
Enter the message of no more than 20 char when done, press <Enter>: Hello
The ciphertext: Khoor
clear screen
change font
0/16


emulator screen (99x25 chars)
Welcome to the shifting program
if you want to encrypt, type E or type D for decryption E
Enter the shifting key (a single digit from 1 to 9): 4
Enter the message of no more than 20 char when done, press <Enter>: ATTACKATONCE
The ciphertext: EXXEGOEXSRGI
clear screen
change font
0/16


SCR emulator screen (80x25 chars)
Welcome to the shifting program
if you want to encrypt, type E or type D for decryption D
Enter the shifting key (a single digit from 1 to 9): 3
Enter the message of no more than 20 char when done, press <Enter>: Khoor
The plaintext: Hello
clear screen
change font
0/16

The Code:
#Hope that we will be able to got a full mark

```
; Welcome to the shift cipher program
data segment
    ; add your data here!
    Enter   db 0AH,0DH,"$"
    welcome db "Welcome to the shifting program $"
    choice  db "if you want to encrypt, type E or type D for decryption $"
    Error_M db  "That is an illegal character. Please try again :) "
    wrong   db   0AH,0DH,"Wrong choice please try again $"
    key     db 0ah,0dh,"Enter the shifting key (a single digit from 1 to 9) :  $"
    text    db "Enter the message of no more than 20 char when done, press <Enter>: $"
    cipher  db "The ciphertext: $"
    plain   db "The plaintext: $"
    res     db  20        ;MAX NUMBER OF CHARACTERS ALLOWED (20).
            db  ?         ;NUMBER OF CHARACTERS ENTERED BY USER.
            db  20 dup(0) ;CHARACTERS ENTERED BY USER.

ends

stack segment
    dw   128  dup(0)
ends

code segment
```

```
globalization:
        mov ax, data
        mov ds, ax
        mov es, ax
        ret
program_info:
        ;only for displaying the output, input and string so that the user can understand
        call NewL
        mov ah, 9
        mov dx, offset welcome
        int 21h
        call NewL
        ret
choicee:
        mov ah, 9
        mov dx, offset choice
        int 21h
        mov AH,1
        int 21h
        cmp al,'E'
        JE Read ; if staetment
        cmp al,'D'
        JE Read ; if else
        call NewL
        mov ah, 9
        mov dx, offset Error_M
        int 21h
        call NewL
        mov ah, 9
        mov dx, offset Wrong
        int 21h
        JMP choicee;to enter a valid choice which is the else staetment
        ret
```

```
Read:

        mov [3100h],al ; save the choice as the question wants
        mov ah, 9
        mov dx, offset key
        int 21h
        mov ah,1
        int 21h
        sub al,30h
        mov [3101h],al
        cmp [3100h], 'D'
        JE Decode
        call NewL
        mov ah,9
        mov dx, offset text
        int 21h
        mov ah, 0Ah ;SERVICE TO CAPTURE STRING FROM KEYBOARD.
        mov dx, offset res
        mov [3102h],dx
        int 21h
        call NewL
        call encrypt
        ret

Encrypt:
        mov si, offset res
        add si, 1
        mov ch, 0
        mov cl,[si]
        ;push cx

        call Repeat
        ret     ; we will use it in the decryption
```

```
Repeat:
        inc si
        mov al,[si]         ;GET THE CHAR FROM BUFFER
        add al,[3101H]      ;ADD KEY
        mov [si-2],al       ;STORE IT TO THE BEGINING OF THE BUFFER
        loop Repeat         ;REPEAT
        mov [si],'$'        ;ADD END OF STRING TO THE LAST 2 LOCATIONS
        mov [si-1],'$'
        mov ah, 09h
        mov ah,9
        mov ah,09h
            mov bl,9
            mov cx, 14  ; mov to cx number of char
            int 10h
        mov dx,offset cipher
        int 21h
        mov ah,9
        mov dx, offset res   ;DISPLAY THE STRING AFTER ENCRYPTION
        int 21h
        ret

Decode:
        call NewL
        mov ah,9
        mov dx, offset text
        int 21h
        mov ah, 0Ah ;SERVICE TO CAPTURE STRING FROM KEYBOARD.
        mov dx, offset res
        mov [3102h],dx
        int 21h
        call NewL
        mov si, offset res
        add si, 1
        mov ch, 0
        mov cl,[si]
        call looping
        ret ;LOAD CHAR COUNT INTO CX
```

```
Looping:
        inc si
        mov al,[si]         ;GET THE CHAR FROM BUFFER
        sub al,[3101H]      ;ADD KEY
        mov [si-2],al       ;STORE IT TO THE BEGINING OF THE BUFFER
        loop Looping          ;REPEAT
        mov [si],'$'        ;ADD END OF STRING TO THE LAST 2 LOCATIONS
        mov [si-1],'$'
        mov ah, 09h
        mov ah,9
        mov ah,09h
            mov bl,9
            mov cx, 14  ; mov to cx number of char
            int 10h
        mov dx,offset plain
        int 21h
        mov ah,9
        mov dx, offset res   ;DISPLAY THE STRING AFTER ENCRYPTION
        int 21h
        ret


NewL:       mov ah,9
        mov dx,offset Enter
        int 21h
        ret
```

```
ENDD:
            mov AH,4ch
            int 21h
start:
      call globalization ;to make code segment local
      call program_info  ;welcoming message for the code
      call choicee
      call ENDD
ends
end start
```