REPORT

SECURITY LOCK

SALMA YOUSRY KAMEL

ID: 7353

GROUPE 3 – SECTION 2

The program contains the identification number and the corresponding password of 20 employees.
Every time you enter an ID it loop over the array that contains the ID's and check the corresponding password

Code:

```
;;; this maro is copied from emu8086.inc ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; this macro prints a char in AL and advances
; the current cursor position:
PUTC    MACRO   char
        PUSH    AX
        MOV     AL, char
        MOV     AH, 0Eh
        INT     10h
        POP     AX
ENDM
org 100h
jmp start1
; define variables:

msg0 db "                    Welcome to the security lock!",0Dh,0Ah,'$'
msg1 db 0Dh,0Ah, 'Please enter your identification number: $'
msg2 db 0Dh,0Ah, 'Please enter your password: $'
msg3 db 0Dh,0Ah," ALLOWED$"
msg4 db 0Dh,0Ah,"DENIED$"
msg5 db 0Dh,0Ah,"Incorrect Identification Number$ "
msg6 db 0Dh,0Ah,"Incorrect Password$"
msg7 db 0Dh,0Ah,"press 1 to exit or ENTER check another employee:$"

IDARR  DW 8000,8001,8002,8003,8004,8005,8006,8007,8008,8009,8010,8011,8012,8013,8014,8015,8016,8017,8018,8019
PASS db 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,1,2,3,4

; identification number and password
id dw ?
password db ?
count dw 0
```

```
start1:

mov dx,offset msg0
mov ah, 9
int 21h

start2:


lea dx,msg1
mov ah, 09h
int 21h

call scan_num
; store identification number:
mov id, cx

; new line:
putc 0Dh
putc 0Ah

lea dx, msg2
mov ah, 09h     ; output string at ds:dx
int 21h
call scan_num
; store password:
;cl because password is byte not word
mov password, cl
; new line:
putc 0Dh
putc 0Ah
mov cx,20
mov bx,0
;count keep track of the ID in the array as well as the corresponding password
mov count,0
checkNUM:
   mov dx,id
   cmp dx,IDARR[bx]
   je  checkPassword
   ;Array ID is of 16 bit word each element occupies 2 bytes of memory
```

```
    ;so we increment twice
    inc bx
    inc bx
    inc count
    loop checkNUM
    jmp  incorrectID

checkPassword:
    mov dl,password
    mov bx,count
    cmp dl,PASS[bx]
    je  allowed
    jmp incorrectPASS
allowed:
      lea dx, msg3
      mov ah, 09h
      int 21h
      jmp finish

incorrectID:
      lea dx,msg5
      mov ah, 09h
      int 21h
      jmp denied
incorrectPASS:
      lea dx, msg6
      mov ah, 09h
      int 21h
      jmp denied


denied:
      lea dx, msg4
      mov ah, 09h
      int 21h

finish:

    ; new line:
    putc 0Dh
```

```
   putc 0Ah

   lea dx,msg7
   mov ah,09h
   int 21h
   call SCAN_NUM
   cmp cx,1
   jne start2
   je  exit

exit:
   mov ah,4ch
   int 21h


SCAN_NUM       PROC    NEAR
      PUSH    DX
      PUSH    AX
      PUSH    SI

      MOV     CX, 0

      ; reset flag:
      MOV     CS:make_minus, 0

next_digit:

      ; get char from keyboard
      ; into AL:
      MOV     AH, 00h
      INT     16h
      ; and print it:
      MOV     AH, 0Eh
      INT     10h

      ; check for MINUS:
      CMP     AL, '-'
      JE      set_minus

      ; check for ENTER key:
```

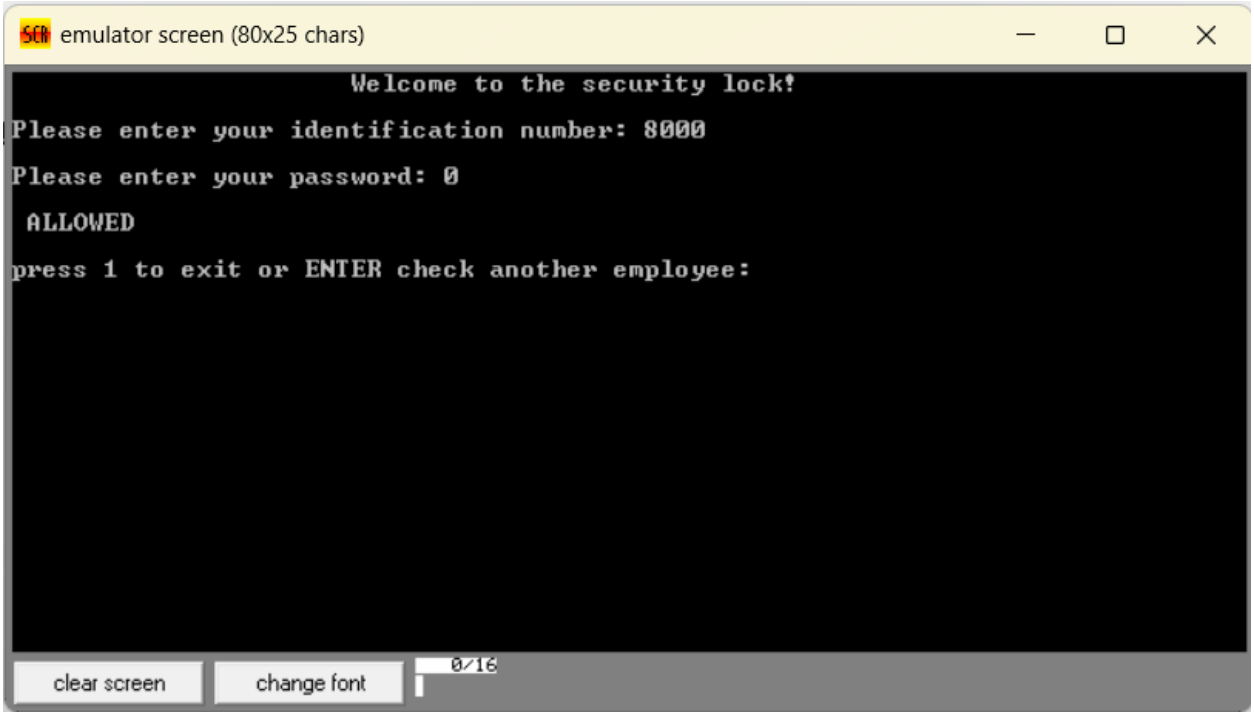
```
        CMP     AL, 0Dh  ; carriage return?
        JNE     not_cr
        JMP     stop_input
not_cr:


        CMP     AL, 8                   ; 'BACKSPACE' pressed?
        JNE     backspace_checked
        MOV     DX, 0                   ; remove last digit by
        MOV     AX, CX                  ; division:
        DIV     CS:ten                  ; AX = DX:AX / 10 (DX-rem).
        MOV     CX, AX
        PUTC    ' '                     ; clear position.
        PUTC    8                       ; backspace again.
        JMP     next_digit
backspace_checked:


        ; allow only digits:
        CMP     AL, '0'
        JAE     ok_AE_0
        JMP     remove_not_digit
ok_AE_0:
        CMP     AL, '9'
        JBE     ok_digit
remove_not_digit:
        PUTC    8       ; backspace.
        PUTC    ' '     ; clear last entered not digit.
        PUTC    8       ; backspace again.
        JMP     next_digit ; wait for next input.
ok_digit:


        ; multiply CX by 10 (first time the result is zero)
        PUSH    AX
        MOV     AX, CX
        MUL     CS:ten                  ; DX:AX = AX*10
        MOV     CX, AX
        POP     AX
```

```
        ; check if the number is too big
        ; (result should be 16 bits)
        CMP     DX, 0
        JNE     too_big

        ; convert from ASCII code:
        SUB     AL, 30h

        ; add AL to CX:
        MOV     AH, 0
        MOV     DX, CX      ; backup, in case the result will be too big.
        ADD     CX, AX
        JC      too_big2    ; jump if the number is too big.

        JMP     next_digit

set_minus:
        MOV     CS:make_minus, 1
        JMP     next_digit

too_big2:
        MOV     CX, DX      ; restore the backuped value before add.
        MOV     DX, 0       ; DX was zero before backup!
too_big:
        MOV     AX, CX
        DIV     CS:ten  ; reverse last DX:AX = AX*10, make AX = DX:AX / 10
        MOV     CX, AX
        PUTC    8       ; backspace.
        PUTC    ' '     ; clear last entered digit.
        PUTC    8       ; backspace again.
        JMP     next_digit ; wait for Enter/Backspace.


stop_input:
        ; check flag:
        CMP     CS:make_minus, 0
        JE      not_minus
        NEG     CX
not_minus:
```

```
        POP     SI
        POP     AX
        POP     DX
        RET
make_minus      DB      ?       ; used as a flag.
SCAN_NUM        ENDP
ten             DW      10      ; used as multiplier/divider by SCAN_NUM & PRINT_NUM_UNS
```

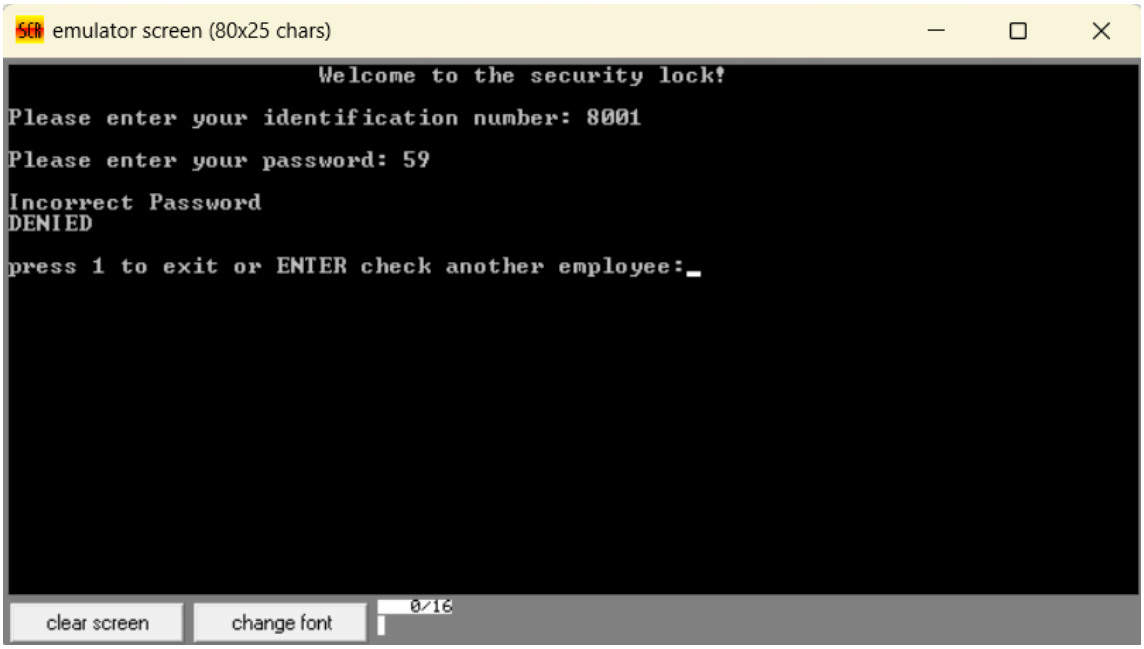
Sample runs:

First case if you enter the correct ID and password it prints "Allowed"



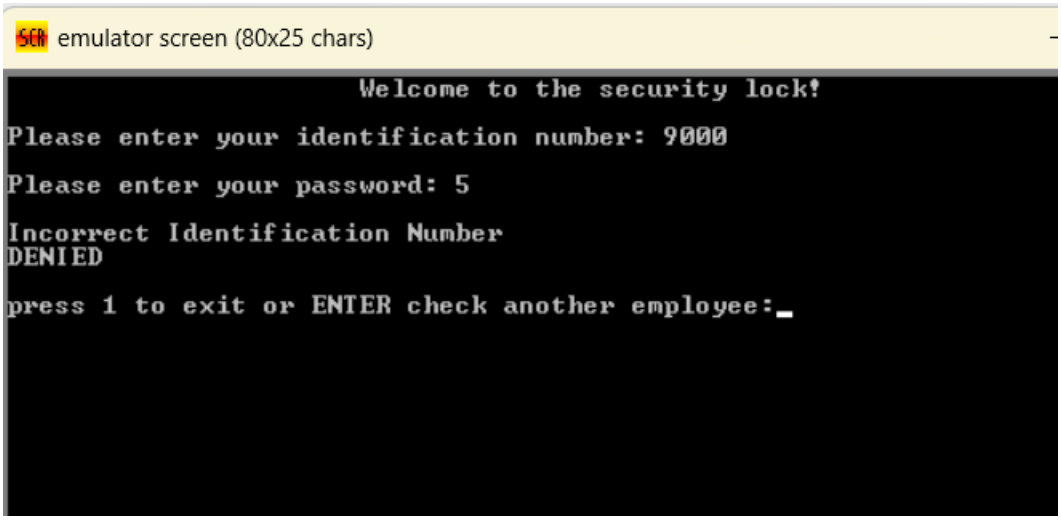
Second case if you enter the correct ID and wrong password it prints

"Incorrect password and DENIED"

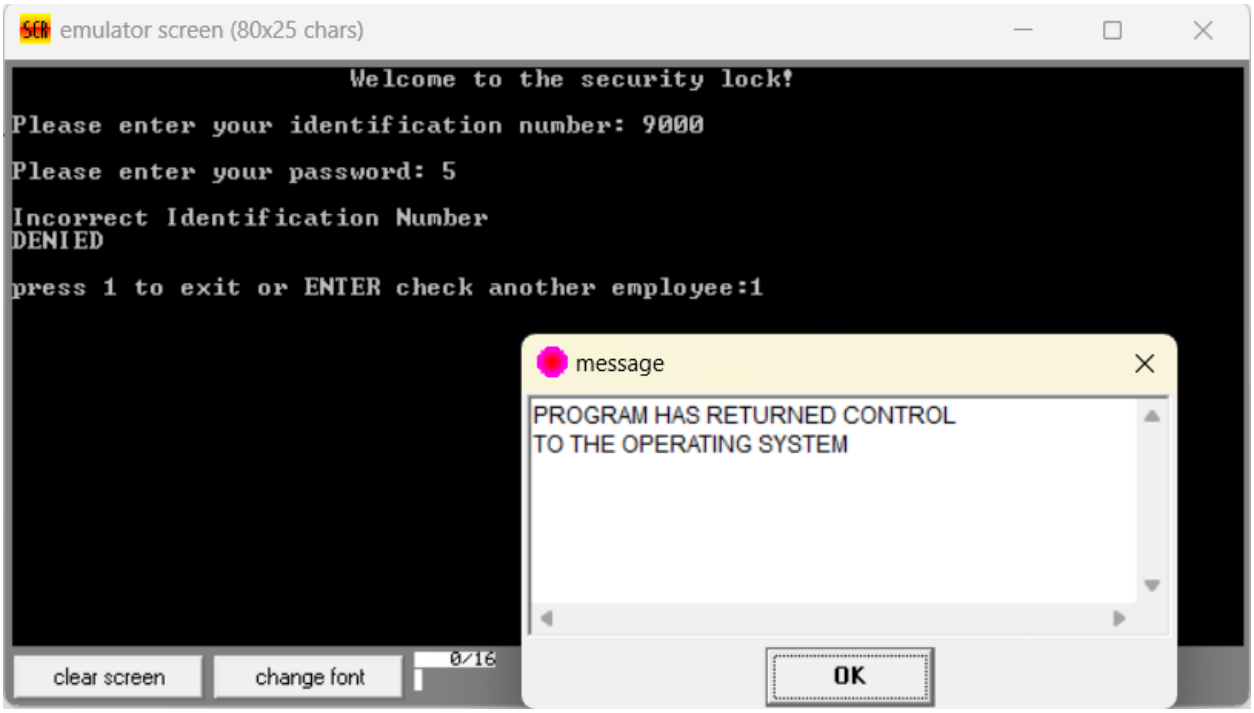


Third case if you enter the wrong ID it prints

"Incorrect identification number and DENIED"



If you want to exit you press 1 or Enter to check another employee


emulator screen (80x25 chars)
Welcome to the security lock!
Please enter your identification number: 9000
Please enter your password: 5
Incorrect Identification Number
DENIED
press 1 to exit or ENTER check another employee:1
message
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
OK
clear screen
change font
0/16