Date 13/10/2022

1) design & develop an ALP to search a key element 'x' using BS.

Sol:
```
.MODEL SMALL
;MACRO TO DISPLAY THE MESSAGE
DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H

ENDM
.DATA
LIST DB 01H, 05H, 07H, 10H, 12H, 14H
NUMBER EQU ($-LIST)
KEY DB 012H
MSG1 DB 0DH, 0AH, "ELEMENT found in the li
MSG2 DB 0DH, 0AH, "Search failed!! elemen
                   not found in list $"
.CODE
START: MOV AX, @DATA
       mov DS, AX
       mov CH, Number-1      ;high value
       mov CL, 00H           ;Low value
AGAIN: MOV SI, OFFset list
       XOR AX, AX
       CMP CL, CH
       JE NEXT
       JNC FAILED
NEXT:  MOV AL, CL
       ADD AL, CH
       SHR AL, 01H           ;% by 2
       mov BL, AL
```

```
        XOR   DH, AH
        MOV   BP, AX
        MOV   AL, DS:[BP][SI]
        CMP   AL, KEY
        JE    SUCCESS
        JC    INCLOW.
        MOV   CH, BL                ; IF KEY>A[I]
        DEC   CH                      SHIFT RIGHT
        JMP   AGAIN
INCLOW: MOV   CL, BL
        INC   CL
        JMP   AGAIN
SUCCESS: DISPLAY MSG1
        JMP   FINAL
FAILED: DISPLAY MSG2
FINAL:  MOV   AH, 4CH
        INT   21H
        END
```

output:

element found in the list

Bubble sort program

Design flowchart to sort the 16-bit numbers using Bubble sort

```
.Model Small
DISPLAY MACRO msg
  LEA DX, msg
  mov AH, 09H
  int 21H
ENDM
.DATA
LIST DB 02H, 08H, 34H, 0F4H, 09H, 05H
number Equ $-List
MSG1 DB 0DH, 0AH, "1 >>> SORT in Ascending order"
MSG2 DB 0DH, 0AH, "2 >> SORT in Descending order"
MSG3 DB 0DH, 0AH, "3>> exit"
msg4 DB 0DH, 0AH, "ENTER Your choice" :: $"
MSG5 DB 0DH, 0AH, "INVALID choice entered.. $
.CODE
START: mov AX, @data
       mov DS, AX
       mov CH, Number-1
       DISPLAY msg1
       Display msg2
       Display msg3
       Display msg4
       mov AH, 01H
       Int 21h
       Sub AL, 30H
       cmp AL, 01H
       JE ASCSORT
       cmp AL, 02H
```

```
IE DESSORT
CMP AL,03H
JE FINAL
DISPLAY MSGS
JMP FINAL
ASCCORT : MOV BL,00H
AGAIN : MOV SI,OFFSET LIST
        MOV CL,00H
        MOV BH,CH
        SUB BH,BL
NPASS : CMP CL,BH
        INC NEXT
        MOV AL,[SI]
        MOV BP,01H
        CMP AL,DS:[BP][SI]
        JC NOPE
        XCHG AL,[SI+1]
        XCHG [SI],AL
NOPE : INC CL
       INC SI
       JMP NPASS
NEXT : INC BL
       CMP BL,CH
       JC Again
       jmp Final.
DESORT : MOV BL,00H
AGAIN : MOV SI,OFFSET LIST
```

```
MOV CL, 00H
MOV BH, CH
SUB BH, BL
NPASS1 : CMP CL, BH
         JNC NEXT
         MOV BP, 01H
         CMP AL, DS[BP][SI]
         JNC _NOPE1
         XCHG AL, [SI+1]
         XCHG [SI], AL
_NOPE1 : INC CL
         INC SI
         JMP NPASS1
NEXT : INC BL
       CMP BL, CH
       JC AGAIN1
FINAL : MOV AH, 4CH
        INT 21H
END
```

III. design a ALP to read a alphanumeric char & display its hex value.

```
.model small
.data
msg1 db 0dh,0ah,"Enter alphanumeric char
     ...acter :$"
res db 02 dup(0)
.code
mov ax,@data
mov ds,ax
lea dx, msg1
call disp
mov ah,01h
int 21h
mov bl,al
mov cl,4
shr al,cl
cmp al,0ah
jc digit
Add AL,07H
digit: add al,30H
       mov res,al
       and bl,0fh
       cmp bl,0ah
       jc digit1
       add bl,07h
digit1: add bl,30h
       mov res+1,bl
```

```
mov ah, 00h
mov al, 03h        ; text mode
int 10h

mov ah, 02h        ; set curior pos
mov bh, 00h        ; page numbeq
mov dh, 0ch        ; Row (00 is top
mov dl, 28h        ; column val.
int 10h

mov rest+2, '$'
lea dx, res
call disp
mov ah, 4ch
int 21h

disp proc near
mov ah, 09h
int 21h
ret
disp endp
end
```

Pallindrome

4) Write ALP to check whether given string is

.model small

(5) display macro msg
lea dx, msg
mov AH, 09H
Int 21H
ENDM

.data
msg1 DB 0DH, 0AH, "enter string: $"
msg2 DB 0DH, 0AH, "Reverse string: $"
msg3 DB 0DH, 0AH, "Input string is pallindrome$"
msg4 DB 0DH, 0AH, "not a pallindrome. $"
String DB 80H DUP(?)
rstring DB 80H DUP(?)

.code
start: mov ax, @data
mov ds, ax
Display msg1
mov SI, offset string
XOR CL, CL
Again: mov AH, 01H
Int 21H
Cmp AL, 0DH ; Bmsce
ASCI value of enter key
JE NEXT
mov [SI], AL
INC SI
INC CL
imp again
next: mov [SI], Byte PTR '$'
DEC SI
mov CH, CL ; reverse the string & store
rstring

```
        mov DI, offset Rstring
Back:   mov Al, [SI]
        mov [DI], Al
        DEC SI
        INC DI
        DEC CH
        JNZ Back
        mov [DI], byte ptr '$'
        display msg2
        Display rstring
        mov SI, offset string
        mov DI, offset Rstring
AG:     mov Al, [SI]
        Cmp Al, [DI]
        JNE Fail
        INC SI
        INC DI
        DEC CX
        JZ SUCCESS
        Jmp AG
FAIL:   Display msg4
        Jmp Final
SUCCESS: Display msg3
FINAL:  mov AH, 4Ch
        int 21h
```

String comparing

read two strings & compare using MP STRING

```
.model small
display macro msg
    lea dx, msg
    mov AH, 09H
    Int 21H.
endm
.Data
msg1 db 0dH, 0ah, "enter first string: $"
msg2 db 0dH, 0ah, "enter second string: $"
msg3 db 0dH, 0ah, "Length of first string: $
msg4 db 0dH, 0ah, "Length of second string:
msg5 db 0dH, 0ah, "strings are equal--$"
msg6 db 0dH, 0ah, "strings are not equal--$
string1 db 80h dup(?)
string2 db 80h dup(?)
.code
start: mov ax, @data
    mov ds, ax
    display msg1
    mov si, offset string1
    call readstr
    mov bl, cl.
    display msg2
    mov si, offset string2
    call readstr
    push Bx
    push cx
```

```
display msg3
mov Al, Bl
call Len_dis
display msg4
mov Al, cl
call Len_dis
pop cx
pop Bx
cmp Cl, Bl
JNE fail
mov SI, OFFset string 1
mov DI, offset string 2
CLD
CHK : mov Al, [SI]
      mov Al, [DI]
      JNE FAil
      INC si
      Inc di
      dec cl
      JNZ CHK
      display msg5
      Jmp final
Len_dis proc near
  xor AH, AH
  Add Al, 00H
  AAm
  Add Ax, 3030H
  mov BH, Al
```

```
mov dl, 0Ah
mov AH, 02H
int 21H
Ret
Len_dis endp
readstr proc near
  XOR cl, cl
Back: mov Al, 01H
      Int 21H
      cmp Al, 0DH
      JE Finish
      mov[si], Al
      inc si
      inc cl
      Jmp Back
Finish: mov[si], Byte ptr $
  ret
readstr endp
fail: display msg6
final: mov AH, 4Ch
       int 21h
end start
```

6 develop ALP to compute nCr

```
.model small
.data
n dw 4
r dw 2
ncr dw 0
.code
mov ax,@data
mov ds,ax
mov ax,n
mov bx,r
call ncrpro
call disp
jmp final
ncrpro proc near
    cmp ax,bx
    je res1
    cmp bx,0
    je res1
    cmp bx,1
    je resn
    dec ax
    cmp bx,ax
    je incr
    push ax
    push bx
    call ncrpro
pop bx
pop ax
```

```
dec bx
push ax
push bx
call ncrpro
pop bx
pop ax
ret

re1:inc ncr
ret
incr:inc ncr
resn: add ncr,ax
ret
ncrpro endp
disp proc near
mov bx,ncr
add bx,3030h
mov dl,bh
mov ah,02h
int 21h
mov dl,bl
mov ah, 02h
int 21h
ret
disp endp.
final: mov ah,4ch
       int 21h
       end
```

Q7) Read current time & display in standard

```
.model small
.code
    mov AH, 2CH
    Int 21H
mov Al, CH
AAM
mov BX, AX
call dISP
mov DL, (:)
mov AH, 02H
Int 21H
mov Al, Cl
AAM
```

```
mov al, dh
aam
mov bx, ax
call disp

mov ah, 02h
int 21h
disp proc near
add dl, 30h
mov ah, 02h
int 21h
mov dl, bl
add dl, bl
add dl, 30h
mov ah, 02h
int 21h
ret
disp endp
end
```

Scanned with CamScanner

8/ WAP to simulate a decimal up-counter 0-99

BINCOUNT.ASM

Sln

```
.model small
.code
mov cl, 00
mov AH, 00H
mov Al, 03H
int 10H
Back: mov BH, 00H
      mov DH, 00H
      mov DL, 00H
      mov AH, 02H
      int 10H
mov Al, cl
Add Al, 00H
AAM
Add Ax, 3030H
mov CH, Al
mov dl, AH
mov AH, 02H
int 21H
mov dl, CH
mov AH, 02H
int 21H
```

```
call delay
inc cl
XOR AX, AX
CMP cl, 1000
JNE BACK
JE Lait

delay proc near
  push Ax
  push Bx
  push cx
  mov cx, 00FFH
AG: mov Bx, 0FFFH
AGI: NOP
     dec Bx
     JNZ AGI
     Dec cx
     JNZ AG
     pop cx
     pop Bx
     pop Ax
     ret
delay endp
Lait: mov AH, 4Ch
      int 21H
      End
```

9) Read a pair of input co-ordinates & move cursor to the specified location on the screen

```
.model small
disp macro msg
 lea DX, msg
 mov AH, 09H
  int 21H
ENDM
.data
msg1 row db 02 dup(0)
msg2 COL db 02 dup(0)
msg 1 db 0dh, 0ah, "enter x co-ordinate: $
msg 2 db 0dh, 0ah, "enter y-co-ordinate: $
.code

mov ax, @data
mov ds, ax
disp msg1
mov SI, offset row
call read
disp msg 2
mov SI, offset col
call read
mov    SI, offset row
mov    AH, [SI]
INC    SI
mov    AL, [SI]
SUB    AX, 3030H
AAd
```

```
mov dH, Al          ; row
mov SI, offset col
mov AH, [SI]
INC SI
mov Al, [SI]
Sub Ax, 3030H
AAD
mov dl, al
mov AH, 00
mov Al, 03H
int 10H
mov AH, 02H
int 10H
disp msg3
Jmp final.

Read proc Neal
    mov Cx, 02H
Back: mov AH, 01H
      Int 21H
      mov [SI], Al
      Inc SI
      DEC Cx
JNZ BACK
Ret
Read endp
final: mov AH, 01H
       Int 21H.
```

```
mov AH, 4Ch
Int 21H
end
```

8/ WAP to stimulate a decimal up-count

Creating & deleting files

10) ~~Wap to creating & deleting files~~

```
.model small
display macro msg
lea dx, msg
mov AH, 09H
Int 21H
Endm
.data
msg1 db 0DH, 0AH, "enter the file name for
                    creation: $"
msg2 db 0dh, 0ah, "file created successfully $"
msg3 db 0dh, 0ah, "creation failed $"
msg4 db, 0dh, 0ah, "enter filename for deletion: $"
msg5 db, 0dh, 0ah, "deleted succesfully $"
msg6 db, 0dh, 0ah, "deletion failed $"
Fname1 db 10 dup(0)
Fname2 db 10 dup(0)
.code
mov AX, @data
mov ds, AX
disp msg1
mov SI, 00
BACK1 : mov AH, 01H
        int 21H
cmp AL, 0DH
JE Next1
mov Fname1[SI], AL
INC SI
Jmp back1
```

```
NEXT1: mov FNAME1[SI], '$'
Lea dx, fname1
mov cx, 00
mov AH, 3CH
int 21H
Jc CFail
disp msg2
jmp del
CFail: disp msg3
del: disp msg4
mov SI, 00
back2: mov ah, 01H
       int 21H
       cmp AL, 0DH
       JE next2
       mov fname2[si], AL
       INC SI
       Jmp back2
next2: mov fname2[si], '$'
       Lea dx, fname2
       mov AH, 41H
       int 21H
       JC Dfail
       Disp msg5
       Jmp Last
dfail: disp msg6
Last: mov AH, 4ch
int 21H
end.
```