System Programming
Lab Report

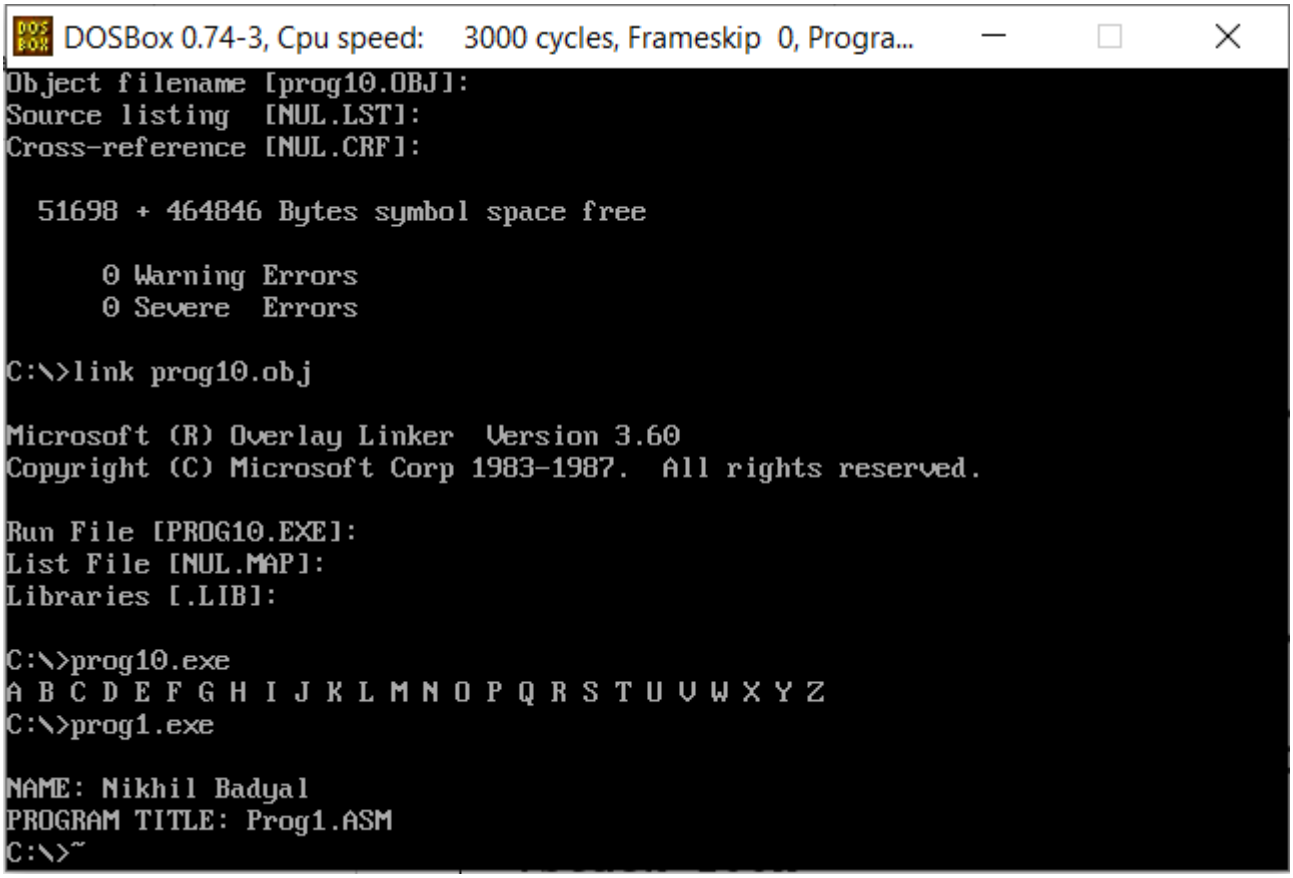
Class – BCSE
Year – 3rd year 1st semester

Nikhil Badyal
Roll No. - 001810501069

1. **Write and test a MASM program to Display your name and program title on the output screen.**

```
.model small
.stack 100h
.data
name1 db 0AH,0DH,'NAME: Nikhil Badyal$'
title1 db 0AH,0DH,'PROGRAM TITLE: A1Q1.ASM$'
.code
print macro msg          ;macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print name1        ;invoking print macro to display name
      print title1       ;invoking print macro to display title

      mov ah, 4ch        ;terminate the program
      int 21h

main endp

end main
```

OUTPUT :

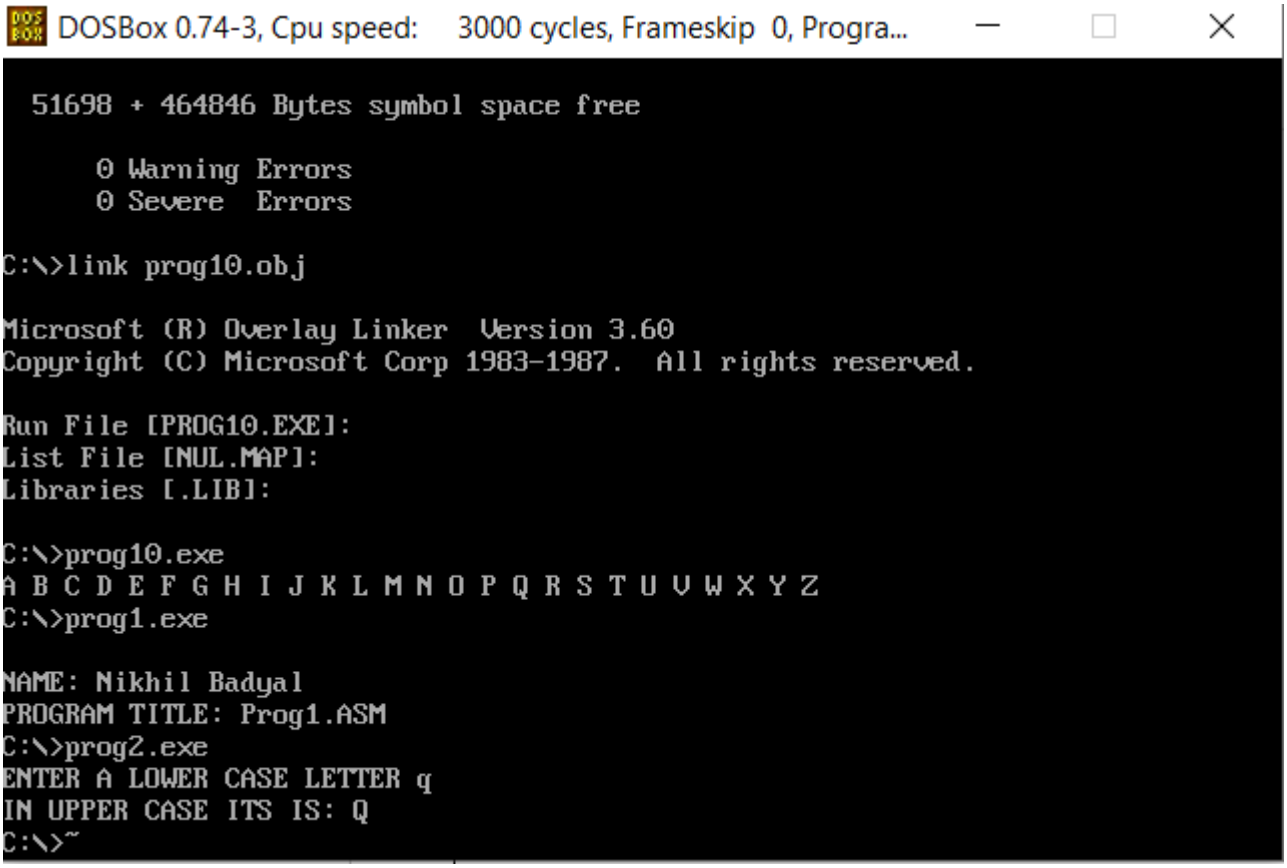


2. Write and test a MASM program to convert a letter from uppercase to lowercase.

```
.model small
.stack 100h
.data
msg1 db 0DH,0AH,'Enter a character: $'
msg2 db 0DH,0AH,'Lower case character: $'
.code
print macro msg         ;macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
```

```
        print msg1
        mov ah,01h  ; read character
        int 21h

        cmp al,'A'
        jl exit
        cmp al,'Z'
        jg exit

        add al,32   ; convert uppercase to lowercase by adding 32 to its ascii

        exit:
        print msg2
        mov dl,al   ; display character
        mov ah,02h
        int 21h
        mov ah, 4ch
        int 21h

main endp

end main
```

OUTPUT :



3. Write and test a MASM program to add two Hexadecimal Numbers.

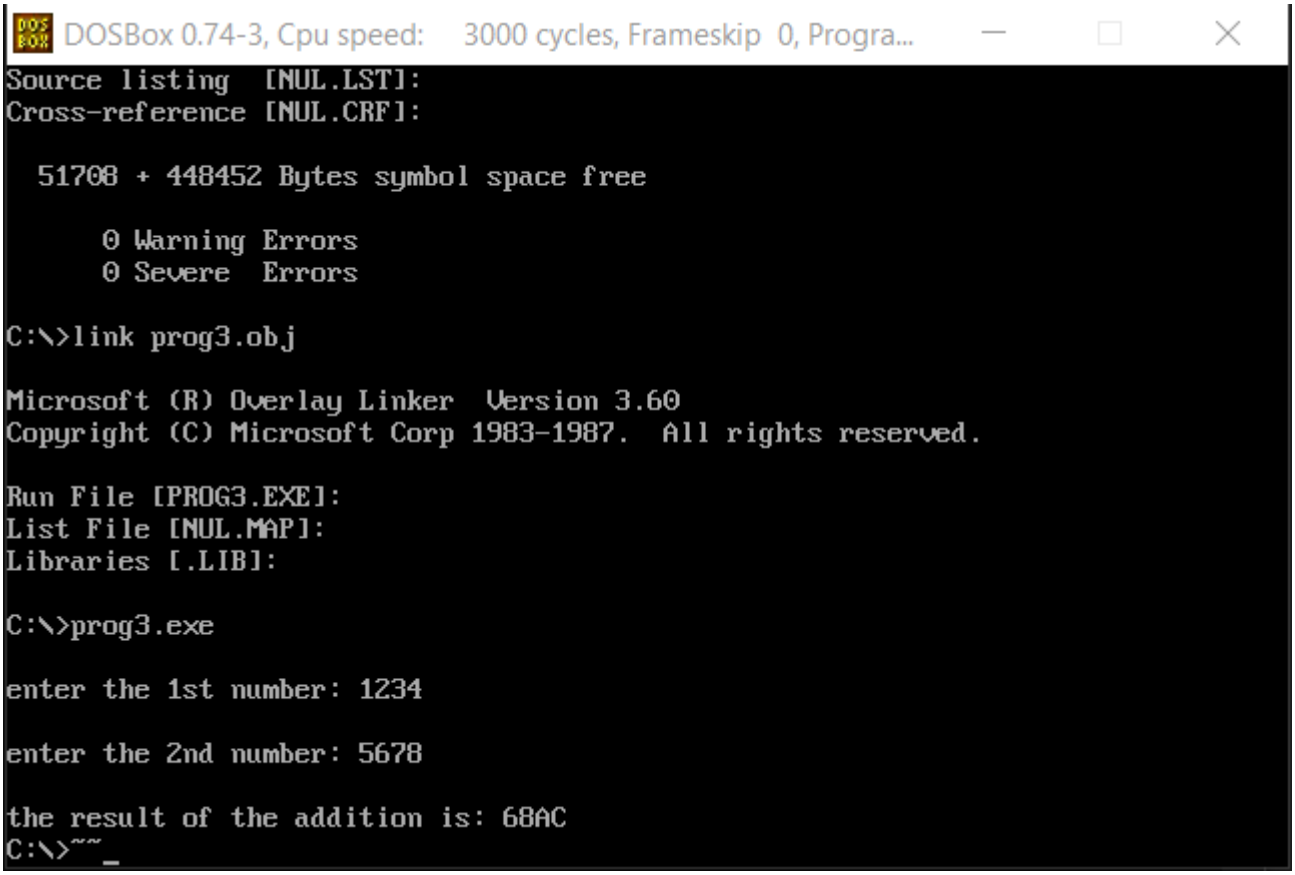
```
.model small
.stack 100h
.data
```

```
      msg1 db 0AH,0DH,'Enter first 16 bit hex number: $'
      msg2 db 0AH,0Dh,'Enter second 16 bit hex number: $'
      msg3 db 0AH,0DH,'Result after adding: $'
.code
print macro msg               ;macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm
main proc
      mov ax, @data           ; initialize data section
      mov ds, ax

      print msg1
      call readhex            ; Read first number
      mov cx, ax
      print msg2
      call readhex            ; Read second number
      print msg3
      add ax,cx               ; add two numbers
```

```
      mov ah, 4cH             ; terminate Program
     int 21H
main endp

readhex proc near
      ; this will input a 16 bit hexadecimal number
      ; output : AX

      push bx
      push cx
      push dx

      xor bx,bx ;initially bx value is equal to 0
      mov cl,4
      mov ah,1  ;for taking input
      int 21h
      input1:
      cmp al,0dh ;compare whether the pressed key is 'enter' or not
      je line1   ;if it is equal to 'enter' then stop taking first value
      cmp al,39h ;compare the input whether it is letter or digit.39h is the ascii
value of 9
      jg letter1
      and al,0fh ;if it is digit then convert it's ascii value to real value by masking
      jmp shift1
        letter1: ;if it is letter then subtract 37h from it to find it's real value
      sub al,37h
      shift1:
      shl bx, cl
      or bl,al ;making 'or' will add the current value with previous value
      int 21h
      jmp input1
      line1:
      mov ax, bx

      pop dx
      pop cx
      pop bx
      ret
readhex endp

writehex proc near
      ; this procedure is to display number in hexadecimal
      ; Input : AX
      push bx
      push cx
      push dx

      mov dx, 0000h
      jnc notcarry
      inc dx
      notcarry:
      mov si, ax
      mov bx, dx         ; Result in reg bx
      mov dh, 2
      l1:   mov ch, 04h     ; Count of digits to be displayed
      mov cl, 04h        ; Count to roll by 4 bits
      l2:   rol bx, cl          ; roll bl so that msb comes to lsb
      mov dl, bl         ; load dl wth data to be displayed
      and dl, 0fH        ; get only lsb
      cmp dl, 09         ; check if digit is 0-9 or letter A-F
      jbe l4
      add dl, 07         ; if letter add 37H else only add 30H
      l4:   add dl, 30H

      mov ah, 02         ; Function 2 under INT 21H (Display character)
      int 21H
      dec ch             ; Decrement Count
      jnz l2
```

```
        dec dh
        cmp dh, 0
        mov bx, si
        jnz l1

        pop dx
        pop cx
        pop bx
        ret
writehex endp

end main
```

OUTPUT :



4. Write and test a MASM program to find the second max and second min from an array.

```
.MODEL SMALL
.STACK 300H
.DATA
ARRAY1 DB 11,22,33,44,55
MSG1 DB 0AH,0DH,'Enter size of the array: $'
MSG2 DB 0AH,0DH,'Second Minimum value in array: $'
MSG3 DB 0AH,0DH,'Second Maximum value in array: $ '
ENDL DB 0AH,0DH,'$'

min dw 99
min2 dw 99
max dw 0
max2 dw 0
SE DB 33H
```

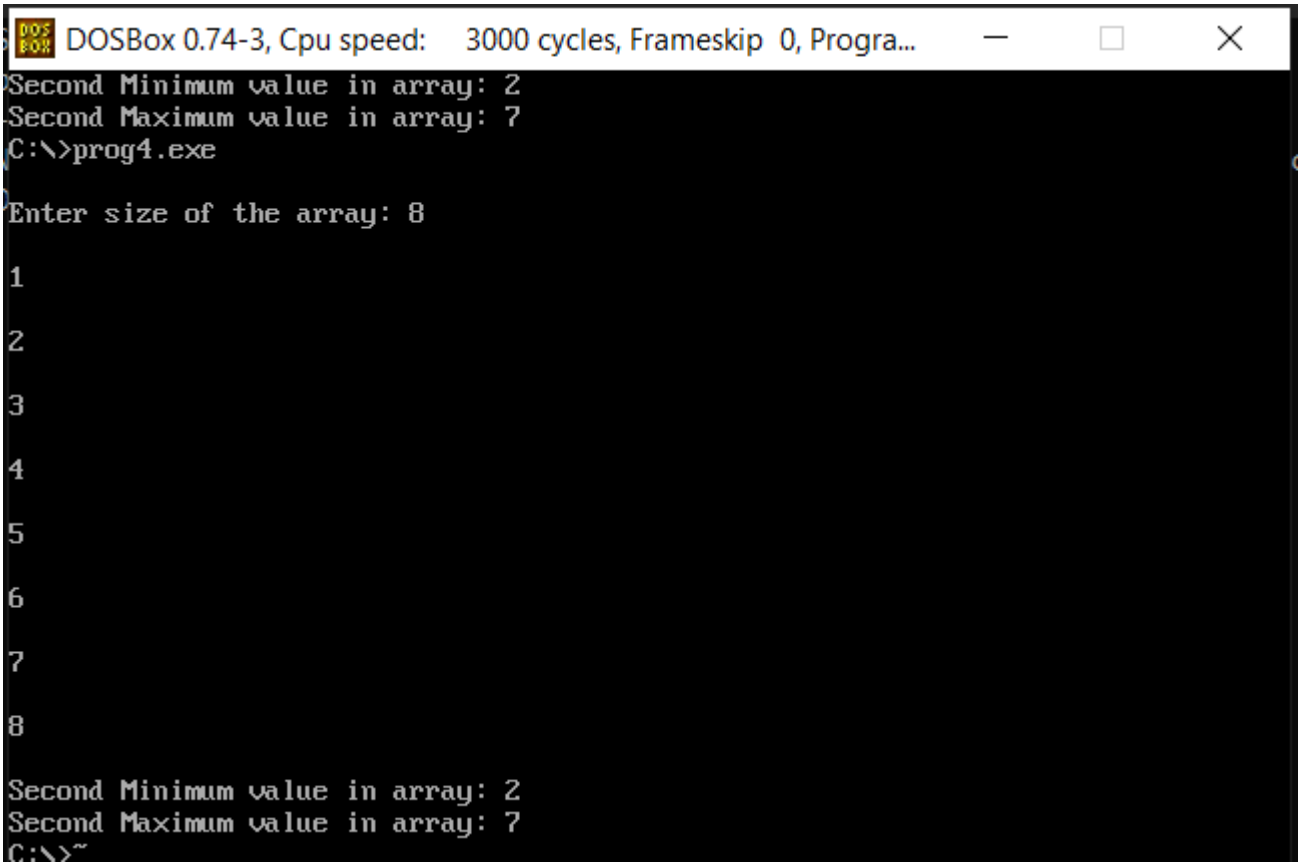
```
COUNT DB 00H

.CODE

PRINT MACRO MSG               ; macro to print a string
      push ax
      push dx
      mov AH, 09H
      lea DX, MSG
      int 21H
      pop dx
      pop ax
ENDM

MAIN PROC
      MOV AX,@DATA
      MOV DS,AX

START:
```

```
      call readnum            ; read the size of array
      mov COUNT, al
      mov cl, COUNT
      mov bx, 00h
      rdnxt:
            PRINT ENDL
            call readnum      ; read an element
            mov ARRAY1[BX],AL ; and storing it in array
            inc BX
      loop rdnxt

      LEA SI, ARRAY1
      call findminmax         ; calling procedure to find min2 and max2

      print msg2
      mov ax, min2            ; second minimum is stored in min2
      call writenum           ; print the result

      print msg3
      mov ax, max2            ; second maximum is stored in max2
      call writenum           ; print the result

      mov ah, 4ch
      int 21h

MAIN ENDP

findminmax PROC
   ; this procedure will print the elements of a given array
   ; input : SI=offset address of the array
   ;       : BX=size of the array
   ; output : none

   PUSH AX                        ; push AX onto the STACK
   PUSH CX                        ; push CX onto the STACK
   PUSH DX                        ; push DX onto the STACK
   push SI
   MOV CX, BX                     ; set CX=BX

   @PRINT_ARRAY:                  ; loop label
     XOR AH, AH                   ; clear AH
     MOV AL, [SI]                 ; set AL=[SI]

      cmp min, ax
      jl notminupdate         ; if min >=  ax
        mov bx, min
        mov min2, bx          ; copy min to min2
      mov min, ax             ; copy ax to min
      jmp update1
      notminupdate:
      cmp min2, ax
      jl update1              ; if min2 >= ax
      cmp ax,min
      je update1              ; and if min2 != ax
      mov min2, ax            ; copy ax to min2
      update1:
      cmp max, ax
      jg notmaxupdate         ; if max <= ax
      mov bx, max
      mov max2, bx            ; copy max to max2
      mov max, ax             ; copy ax to max
      jmp update2
      notmaxupdate:
      cmp max2, ax
```

```
      jg update2              ; if max2  <= ax
      cmp ax, max
      je update2              ; and if max2 != ax
      mov max2, ax            ; copy ax  to max2

      update2:

     MOV AH, 2                    ;  set output function
     MOV DL, 20H                  ;  set DL=20H
     INT 21H                      ;  print a character

     INC SI                       ;  set SI=SI+1
   LOOP @PRINT_ARRAY              ;  jump to label @PRINT_ARRAY while CX!=0

   pop SI
   POP DX                         ;  pop a value from STACK into DX
   POP CX                         ;  pop a value from STACK into CX
   POP AX                         ;  pop a value from STACK into AX

   RET                            ;  return control to the calling procedure
findminmax ENDP


readnum proc near
      ; this procedure is to read a decimal number
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
loopnum:
      mov ah,01h
      int 21h
      cmp al,'0'
      jb skip
      cmp al,'9'
      ja skip
      sub al,'0'
      push ax
      mov ax,bx
      mul cx
      mov bx,ax
      pop ax
      mov ah,00h
      add bx,ax
      jmp loopnum
skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum PROC near
   ; this procedure will display a decimal number
   ; input : AX
   ; output : none

   push bx                        ; push BX onto the STACK
   push cx                        ; push CX onto the STACK
   push dx                        ; push DX onto the STACK

   XOR CX, CX                     ; clear CX
   MOV BX, 10                     ; set BX=10
```

```
   @OUTPUT:                        ; loop label
     XOR DX, DX                    ; clear DX
     DIV BX                        ; divide AX by BX
     PUSH DX                       ; push DX onto the STACK
     INC CX                        ; increment CX
     OR AX, AX                     ; take OR of Ax with AX
   JNE @OUTPUT                     ; jump to label @OUTPUT if ZF=0

   MOV AH, 2                       ; set output function

   @DISPLAY:                       ; loop label
     POP DX                        ; pop a value from STACK to DX
     OR DL, 30H                    ; convert decimal to ascii code
     INT 21H                       ; print a character
   LOOP @DISPLAY                   ; jump to label @DISPLAY if CX!=0

   POP DX                          ; pop a value from STACK into DX
   POP CX                          ; pop a value from STACK into CX
   POP BX                          ; pop a value from STACK into BX

   RET                             ; return control to the calling procedure
writenum ENDP

END MAIN
```

OUTPUT :

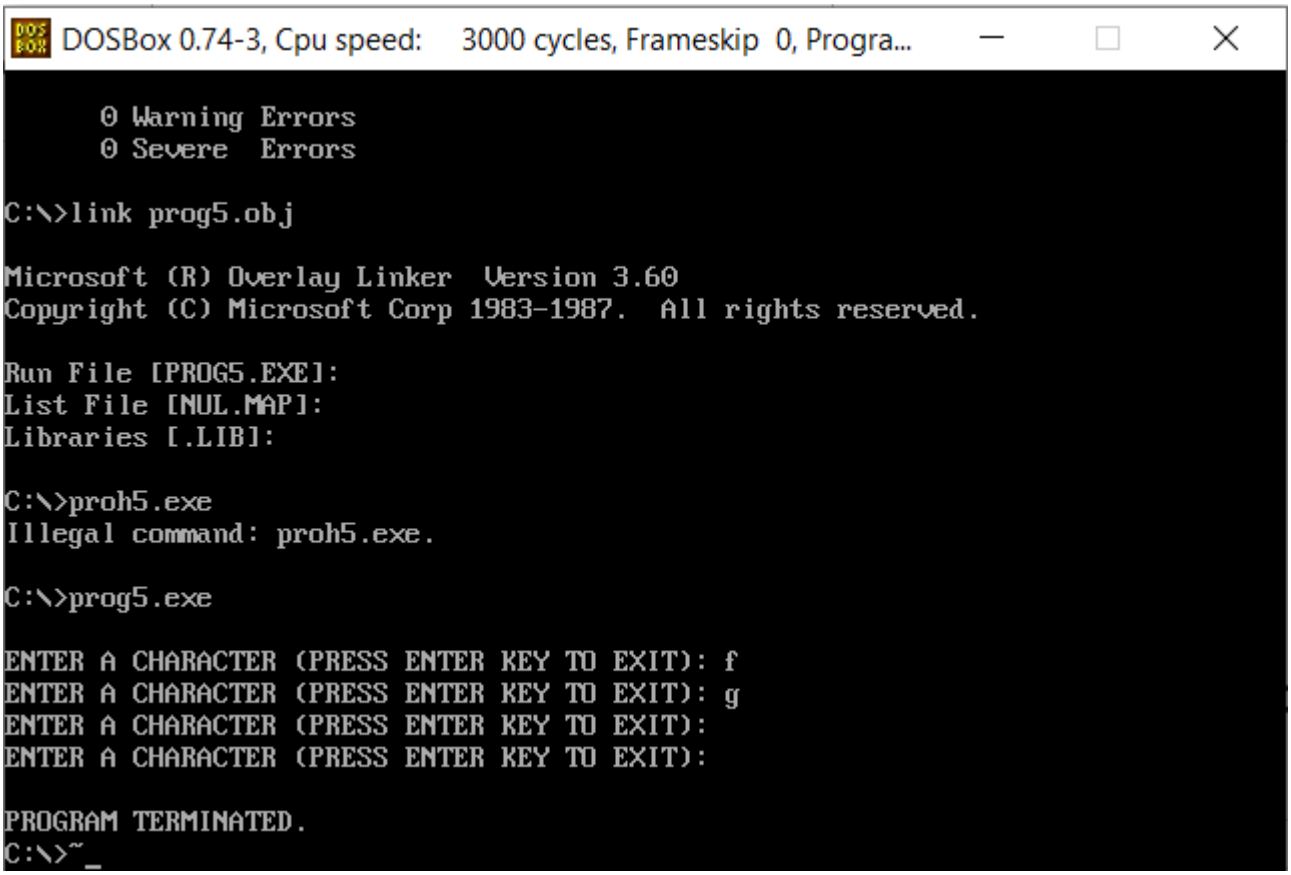


5. Write and test a MASM program to display a terminating message.

```
.model small
.stack 100h
.data
msg1 db 0AH,0DH,'ENTER A CHARACTER (PRESS ENTER KEY TO EXIT):
$' msg2 db 0AH,0DH,'PROGRAM TERMINATED.$'
.code
print macro msg          ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax, @data
      mov ds, ax

      l1:
      print msg1
      mov ah, 01h        ; read a character
      int 21h
```

```
      cmp al,13          ; compare with ASCII value of enter key
      jne l1             ; continue until enter key is not pressed

      print msg2         ; print terminating message

      mov ah,4CH
      int 21h
main endp

end main
```

OUTPUT :

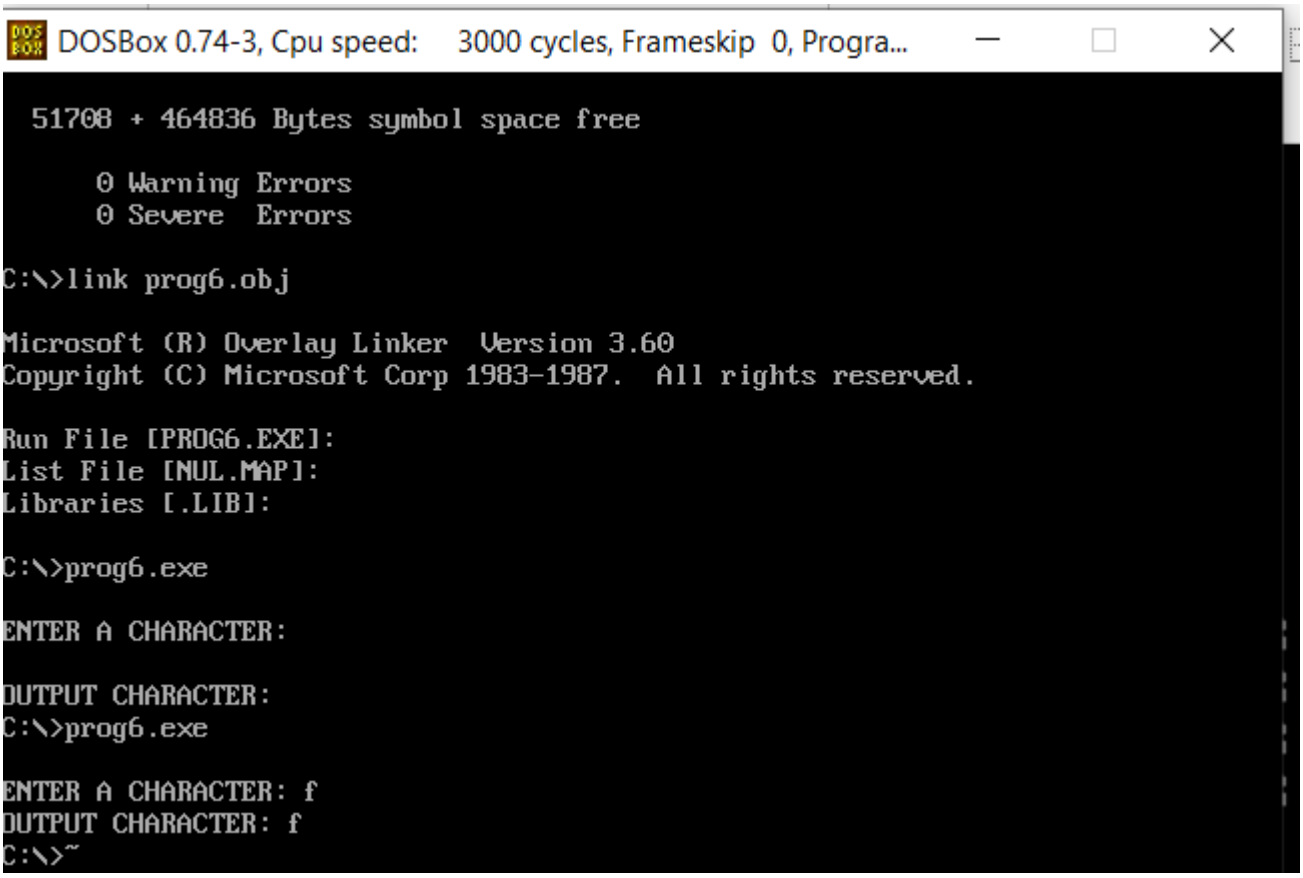


6. Write and test a MASM program to Take a character from keyboard and print it.

```
.model small
.stack 100h
.data
msg1 db 0DH,0AH,'ENTER A CHARACTER: $'
msg2 db 0DH,0AH,'OUTPUT CHARACTER: $'
.code
print macro msg          ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm
```

```
main proc
      mov ax, @data
      mov ds, ax

      print msg1
      mov ah,01h        ;read character
      int 21h

      print msg2
      mov dl, al        ;display character
      mov ah, 02h
      int 21h

      mov ah,4ch
      int 21h
main endp

end main
```

OUTPUT :

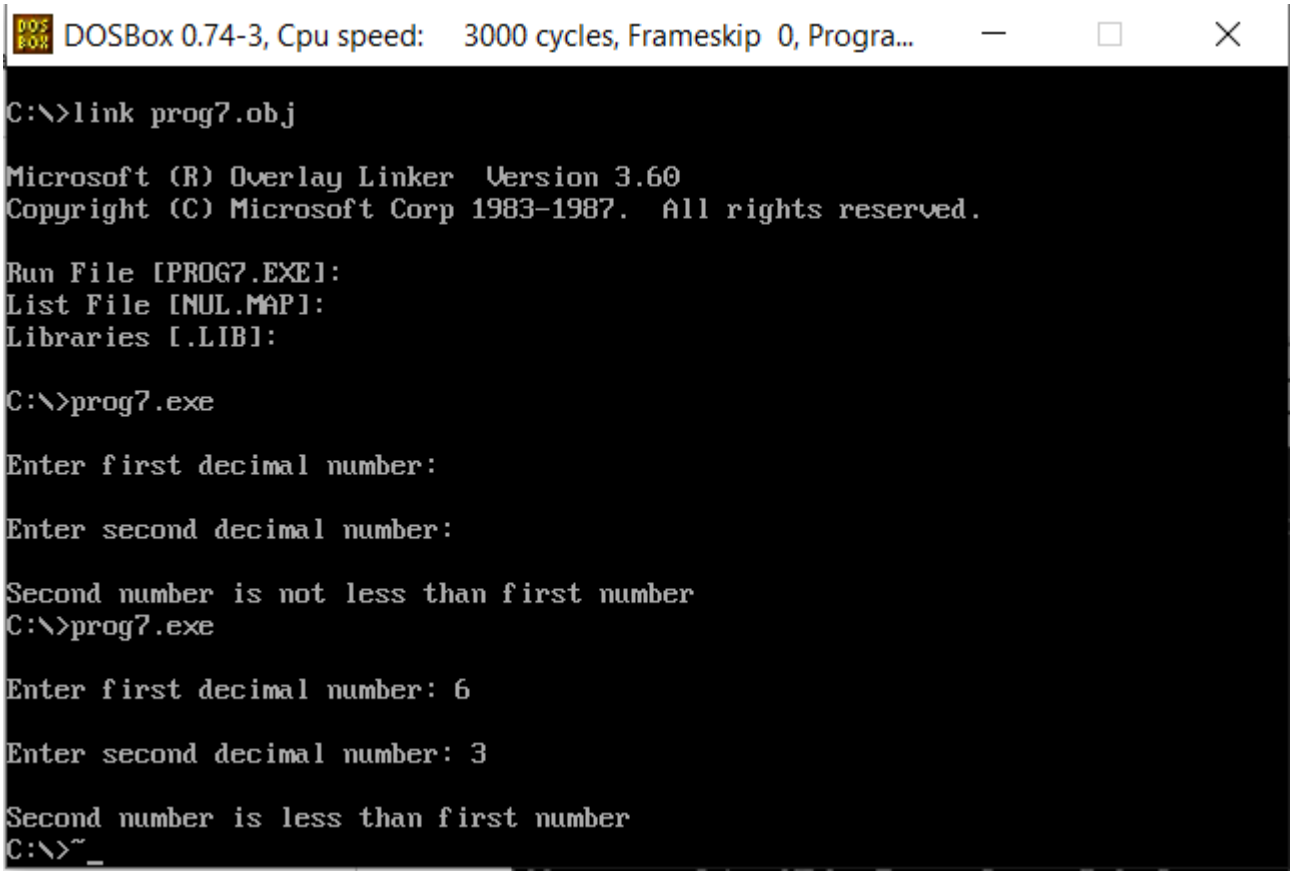


7. Write and test a MASM program to validate second numbers is less than the first.

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'Enter first decimal number: $'
msg2 db 0AH,0DH,'Enter second decimal number: $'
msg3 db 0AH,0DH,'Second number is less than first number$'
msg4 db 0AH,0DH,'Second number is not less than first
number$' .code
print macro msg               ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1
      call readdecimal        ; read first number, value is stored in ax
      mov cx, ax              ; copy first number to cx register

      print msg2
      call readdecimal

      cmp ax,cx               ; compare second with first number
      jl less
      print msg4              ; print message if second number is < first
      jmp exit
      less:                   ; print message if second number is >= first
      print msg3

      exit:
      mov ah, 4ch
      int 21h
main endp

readdecimal proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX

      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
```

```
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readdecimal endp

end main
```

OUTPUT :



8. Write and test a MASM program to find maximum and minimum from an array.

```
.MODEL SMALL
.STACK 300H
```

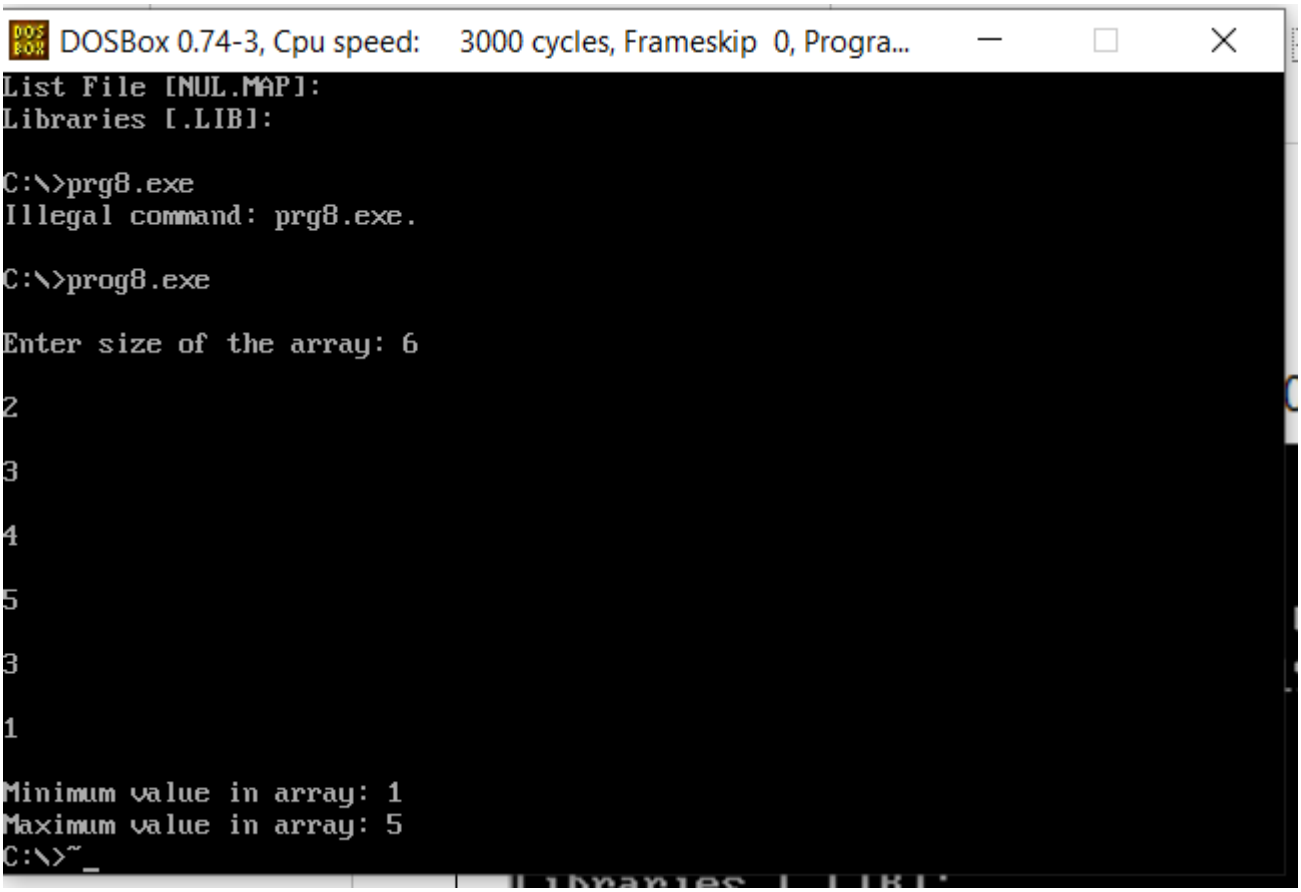
```
.DATA
ARRAY1 DB 11,22,33,44,55
MSG1 DB 0AH,0DH,'Enter size of the array: $'
MSG2 DB 0AH,0DH,'Minimum value in array: $'
MSG3 DB 0AH,0DH,'Maximum value in array: $ '
ENDL DB 0AH,0DH,'$'

min dw 99
max dw 0
SE DB 33H
COUNT DB 00H

.CODE

PRINT MACRO MSG         ; macro to print a string
      push ax
      push dx
      mov AH, 09H
      lea DX, MSG
      int 21H
      pop dx
      pop ax
ENDM

MAIN PROC
```

```
      MOV AX,@DATA
      MOV DS,AX

START:

      PRINT MSG1
      call readnum            ; read the size of array
      mov COUNT, al
      mov cl, COUNT
      mov bx, 00h
      rdnxt:
            PRINT ENDL
            call readnum      ; read each array element
            mov ARRAY1[BX],AL ; storing it in array
            inc BX
      loop rdnxt

      LEA SI, ARRAY1
      call findminmax         ; calling procedure to find min and max

      print msg2
      mov ax, min             ; minimum value is stored in min
      call writenum           ; print the result

      print msg3
      mov ax, max             ; maximum value is stored in max
      call writenum           ; print the result

      mov ah, 4ch
      int 21h

MAIN ENDP

findminmax PROC
   ; this procedure will print the elements of a given array
   ; input : SI=offset address of the array
   ;       : BX=size of the array
   ; output : none

   PUSH AX                        ; push AX onto the STACK
   PUSH CX                        ; push CX onto the STACK
   PUSH DX                        ; push DX onto the STACK
   push SI
   MOV CX, BX                     ; set CX=BX

   @PRINT_ARRAY:                  ; loop label
     XOR AH, AH                   ; clear AH
     MOV AL, [SI]                 ; set AL=[SI]

      cmp min, ax
       jl notminupdate           ; if min >= ax
      mov min, ax                ; copy ax to min
      notminupdate:

      cmp max, ax
       jg notmaxupdate           ; if max <= ax
      mov max, ax                ; copy ax to max
      notmaxupdate:

     MOV AH, 2                    ; set output function
     MOV DL, 20H                  ; set DL=20H
     INT 21H                      ; print a character

     INC SI                       ; set SI=SI+1
   LOOP @PRINT_ARRAY              ; jump to label @PRINT_ARRAY while CX!=0
```

```
   pop SI
   POP DX                          ; pop a value from STACK into DX
   POP CX                          ; pop a value from STACK into CX
   POP AX                          ; pop a value from STACK into AX

   RET                             ; return control to the calling procedure
findminmax ENDP

readnum proc near

      push bx
      push cx
      mov cx,0ah
      mov bx,00h
loopnum:
      mov ah,01h
      int 21h
      cmp al,'0'
      jb skip
      cmp al,'9'
      ja skip
      sub al,'0'
      push ax
      mov ax,bx
      mul cx
      mov bx,ax
      pop ax
      mov ah,00h
      add bx,ax
      jmp loopnum
skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum PROC near
   ; this procedure will display a decimal number
   ; input : AX
   ; output : none

   push bx                         ; push BX onto the STACK
   push cx                         ; push CX onto the STACK
   push dx                         ; push DX onto the STACK

   XOR CX, CX                      ; clear CX
   MOV BX, 10                      ; set BX=10

   @OUTPUT:                        ; loop label
     XOR DX, DX                    ; clear DX
     DIV BX                        ; divide AX by BX
     PUSH DX                       ; push DX onto the STACK
     INC CX                        ; increment CX
     OR AX, AX                     ; take OR of Ax with AX
   JNE @OUTPUT                     ; jump to label @OUTPUT if ZF=0

   MOV AH, 2                       ; set output function

   @DISPLAY:                       ; loop label
     POP DX                        ; pop a value from STACK to DX
     OR DL, 30H                    ; convert decimal to ascii code
     INT 21H                       ; print a character
   LOOP @DISPLAY                   ; jump to label @DISPLAY if CX!=0
```

```
   POP DX                          ; pop a value from STACK into DX
   POP CX                          ; pop a value from STACK into CX
   POP BX                          ; pop a value from STACK into BX

   RET                             ; return control to the calling procedure
writenum ENDP

END MAIN
```

OUTPUT :

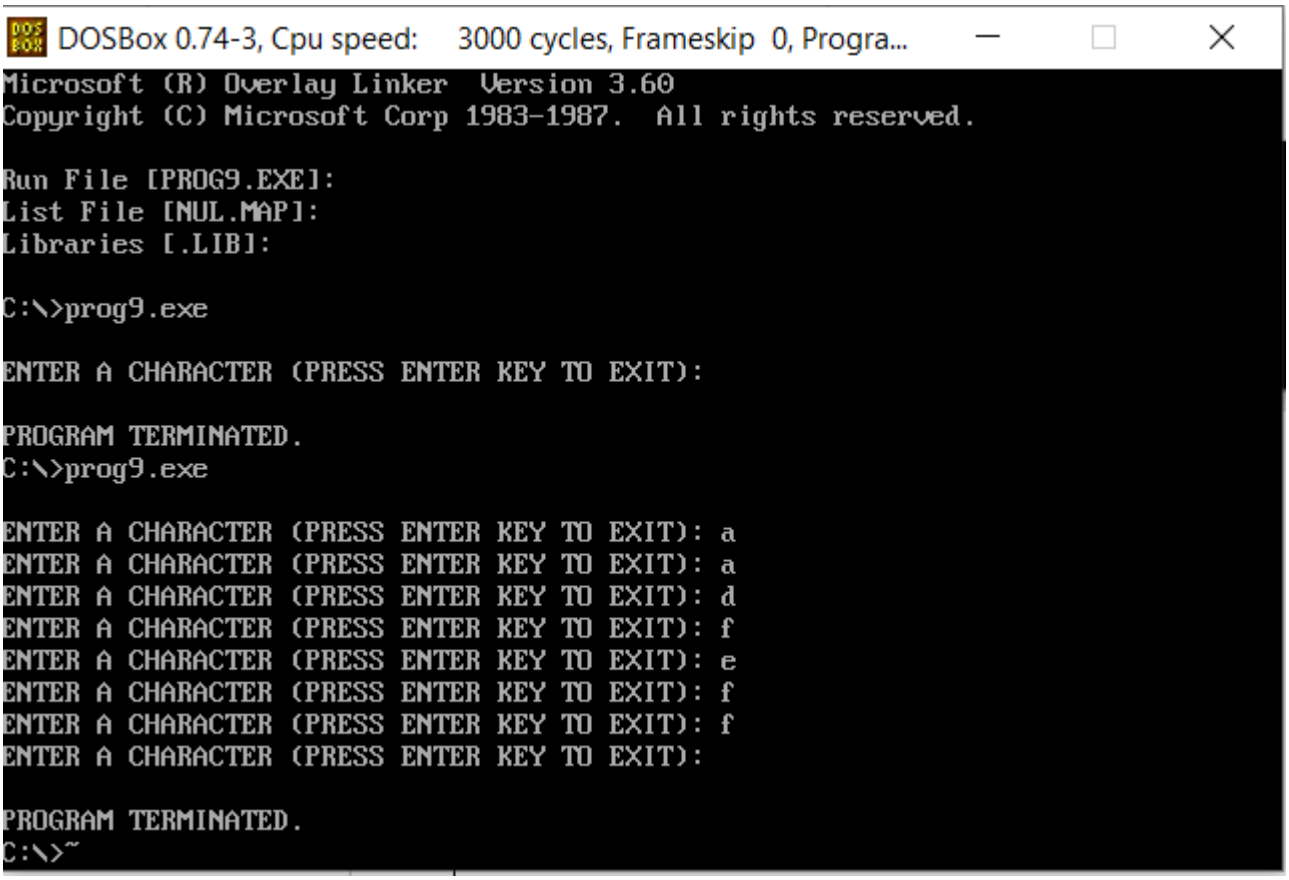


9. Write and test a MASM program to loop until the user decides to quit.

```
.model small
.stack 100h
.data
msg1 db 0AH,0DH,'ENTER A CHARACTER (PRESS ENTER KEY TO EXIT):
$' msg2 db 0AH,0DH,'PROGRAM TERMINATED.$'
.code
print macro msg          ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
```

```
endm

main proc
      mov ax, @data
      mov ds, ax

      l1:
      print msg1
      mov ah, 01h ; read a character
      int 21h

      cmp al,13          ; compare with ASCII value of enter key
      jne l1             ; continue until enter key is not pressed

      print msg2         ; print terminating message

      mov ah,4CH
      int 21h
main endp

end main
```

OUTPUT :



---

10. Write and test a MASM program to print all the characters from A-Z.

```
.model small
.stack 100h
.data
.code
printchar macro char     ; macro to display a character
      push ax
      push dx
```

```
      mov dl,char
      mov ah,02h
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      mov cl,64          ;cl = 64 (ascii value of character just before 'A')
      l1:
      inc cl
      printchar cl       ;print alphabet
      printchar 20h      ;print space (ASCII - 20H)
      cmp cl,'Z'
      jne l1             ;loop until Z occurs

      mov ah, 4CH
      int 21h

main endp
end main
```

OUTPUT :