FARMAGUDI, PONDA GOA

```
1)
%macro print 2
                                        and [size], dword 000fh
     mov eax,sys_out
                                        mov eax, arr
     mov ebx, stdout
                                        mov edx,[size]
     mov ecx,%1
                                        call inputElArr
     mov edx,%2
     int 80h
                                        print p3,p3L
                                        mov eax,arr
     %endmacro
                                        mov edx,[size]
     %macro input 2
                                        call display
     mov eax, sys_in
     mov ebx, stdin
                                        mov eax,1
     mov ecx,%1
                                        mov ebx,0
     mov edx,%2
                                        int 80h
     int 80h
     %endmacro
     section .data
                                        display:
     sys_out equ 4 ;To output
                                        ;; eax should contain address
                                      of array
     sys_in equ 3 ;To Input
                                        ;; edx should contain size of
     stdout equ 1 ;Stdout
                                      array
     stdin equ 2 ;Stdins
                                        ;; prints bytes
                                        mov ecx,0
     p1 db 'ENTER SIZE OF N: '
                                        repeat_display:
     p1L equ $-p1
                                        cmp ecx,edx
                                        ige after_display
     p2 db 'ENTER ELMENTS INTO
YOUR ARRAY:
                                        mov bl,[eax+ecx]
     p2L equ $-p2
                                        pushad
                                        call print_bl
     p3 db 'ARRAY: '
     p3L equ $-p3
                                        mov eax,4
                                        mov ebx, 1
     space db ' '
                                        mov ecx, space
     spaceL equ $-space
                                        mov edx, spaceL
                                        int 80h
     newline db ' '
     newlineL equ $-newline
                                        popad
                                        inc ecx
                                        jmp repeat_display
     section .bss
                                        after_display:
     size resb 9
                                        mov eax,4
     arr resb 10
                                        mov ebx, 1
     dis_buffer resb 3
                                        mov ecx, newline
                                        mov edx, newlineL
     ip_buffer resb 4
                                        int 80h
section _text:
                                        ret
global _start:
  _start:
                                      inputElArr:
  print p1,p1L
  input size,9
                                        ;; eax should contain address
                                      of array
  print p2,p2L
                                        ;; edx should contain size of
  sub [size],dword '0'
                                      array
```

FARMAGUDI, PONDA GOA

```
;; scans bytes
                                         int 80h
  mov ecx,0
  repeat_input:
                                         ret
  cmp ecx,edx
  jge after_input
                                         scan_esi:
                                         ;; esi should contain address
                                      of value to be scanned
  mov ebx,eax
  add ebx,ecx
                                        mov eax, 3
  mov esi,ebx
                                         mov ebx,2
  pushad
                                         mov ecx,ip_buffer
  call scan_esi
                                         mov edx,4
                                         int 80h
  popad
  inc ecx
  jmp repeat_input
                                         mov al,0
  after_input:
                                         mov ecx,0
                                         scan_esi_loop:
                                         cmp ecx, dword 3
  ret
                                         jge after_scan_esi
  print_bl:
  ;; bl should contain value to
                                        cmp byte [ip_buffer+ecx],10
be printed
                                         je after_scan_esi
  mov ecx,3
  mov esi,2
                                         mov bl,10
                                         mul bl
  mov al,bl
                                                              ;al=al*10
                                         mov bl,[ip_buffer+ecx]
  print_bl_loop:
                                         sub b1,'0'
                                                              ;b1=
  mov ah,0
  mov dl,10
                                      digit
  div dl
                                         add al,bl
                                                              ;al=al+bl
  add ah, '0'
  mov [dis_buffer+esi],ah
                                         inc ecx
  dec esi
                                         jmp scan_esi_loop
  loop print_bl_loop
                                        after_scan_esi:
  mov eax,4
                                        mov [esi],al
  mov ebx,1
  mov ecx,dis_buffer
                                         ret
  mov edx, 3
```

OUTPUT

Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop

```
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 8A.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 8A 8A.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./8A
ENTER SIZE OF N: 4
ENTER ELMENTS INTO YOUR ARRAY: 4
15
22
32
ARRAY: 004 015 022 032 lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ _
```

FARMAGUDI, PONDA GOA

```
2)
                                       ;; stores values in pcount and
section .data
pres db "POSITIVE NUMBER COUNT:
                                      ncount variables
                                      mov [pcount], byte 0
                                      mov [ncount], byte 0
prl equ $-pres
nres db "NEGATIVE NUMBER COUNT:
                                      mov ecx,0
                                       repeat_pncount:
nrl equ $-nres
                                      cmp ecx,edx
arr db 4,-10,-15,18,28,26,30,
                                      jge after_pncount
-27,1,-2
newline db 10
                                      mov bl,[eax+ecx]
section .bss
                                      cmp b1,0
pcount resb 1
                                       jl negative
ncount resb 1
                                       jg positive
dis_buffer resb 3
                                      jmp after_np
                                      negative:
%macro SYS_WRITE 2
                                      inc byte[ncount]
mov eax,4
                                       jmp after_np
mov ebx,1
                                      positive:
mov ecx,%1
                                      inc byte[pcount]
mov edx,%2
                                      after_np:
int 80h
                                       inc ecx
%endmacro
                                       jmp repeat_pncount
                                       after_pncount:
section .text
                                       ret
global _start
_start:
                                       print_bl:
mov eax, arr
mov edx, 10
                                       ;; bl should contain value to
call procount
                                      be printed
                                      mov ecx, 3
SYS_WRITE pres,prl
                                      mov esi,2
mov bl,[pcount]
                                      mov al,bl
call print_bl
                                      print_bl_loop:
                                      mov ah,0
SYS_WRITE newline,1
                                      mov dl,10
                                      div dl
                                      add ah, '0'
SYS_WRITE nres,nrl
mov bl,[ncount]
                                      mov [dis_buffer+esi],ah
                                      dec esi
call print_bl
                                       loop print_bl_loop
mov eax,1
mov ebx,0
                                      mov eax,4
int 80h
                                      mov ebx,1
                                      mov ecx,dis_buffer
                                      mov edx,3
                                       int 80h
pncount:
;; eax should contain address
                                       ret
of array
;; edx should contain size of
array
```

FARMAGUDI, PONDA GOA

OUTPUT

Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop

```
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 8B.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 8B 8B.o
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./8B
POSITIVE NUMBER COUNT: 006
NEGATIVE NUMBER COUNT: 004lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
3)
section .data p1 db 'ENTER SIZE OF N: '
                                        _start:
  p1L equ $-p1
                                              SYS_WRITE p1,p1L
                                              SYS_READ size,2
  p2 db 'ENTER ELMENTS INTO
                                              SYS_WRITE p3,p3L
YOUR ARRAY:
  p2L equ $-p2
                                              sub [size],dword '0'
                                              and [size], dword 000fh
  p3 db 'ARRAY: '
                                              mov eax, arr
  p3L equ $-p3
                                              mov edx,[size]
     space db ' '
                                              call input
     ores db "ODD : "
     orl equ $-ores
                                              mov esi,arr
     eres db "EVEN : "
                                              mov edx,[size]
     erl equ $-eres
                                              call oecount
     newline db 10
                                              SYS_WRITE eres,erl
section .bss
                                              mov bl,[ecount]
     arr resb 10
                                              call print_bl
     ocount resb 1
     ecount resb 1
                                              SYS_WRITE newline,1
     dis_buffer resb 3
     ip_buffer resb 4
                                              SYS_WRITE ores,orl
     size resb 4
                                              mov bl,[ocount]
                                              call print_bl
%macro SYS_WRITE 2
     mov eax,4
                                              mov eax,1
     mov ebx, 1
                                              mov ebx, 0
                                              int 80h
     mov ecx,%1
     mov edx,%2
     int 80h
%endmacro
                                        input:
                                              ;; eax should contain
%macro SYS_READ 2
                                        address of array
                                        ;; edx should contain size of array
     mov eax,3
     mov ebx,2
                                              ;; scans bytes
     mov ecx,%1
     mov edx,%2
                                              mov ecx,0
     int 80h
                                        repeat_input:
%endmacro
                                              cmp ecx,edx
                                              jge after_input
section .text
                                              mov ebx, eax
global _start
                                              add ebx,ecx
```

FARMAGUDI, PONDA GOA

```
print_bl_loop:
     mov esi,ebx
     pushad
                                            mov ah,0
     call scan_esi
                                            mov dl, 10
     popad
                                            div dl
                                            add ah, '0'
     inc ecx
                                            mov [dis_buffer+esi],ah
     jmp repeat_input
after_input:
                                            dec esi
                                            loop print_bl_loop
     ret
                                            mov eax,4
                                            mov ebx,1
                                            mov ecx,dis_buffer
oecount:
;; esi should contain address
                                            mov edx.3
of array
                                            int 80h
;; edx should contain size of
                                            ret
array
;; stores values in occunt and
ecount variables
                                      scan_esi:
                                            ;; esi should contain
     mov [ocount],byte 0
                                      address of value to be scanned
     mov [ecount],byte 0
     mov ecx,0
                                            mov eax, 3
repeat_oecount:
                                            mov ebx,2
                                            mov ecx, ip_buffer
     cmp ecx,edx
     jge after_oecount
                                            mov edx,4
                                            int 80h
     mov al,[esi+ecx]
     cbw
                                            mov al,0
     mov b1,2
                                            mov ecx,0
     div bl
     ; a1=a1/2 , ah=a1%2
                                      scan_esi_loop:
     cmp ah,0
                                            cmp ecx, dword 3
                            ;if
     je even
                                            jge after_scan_esi
(ah\%2) == 0
odd:
                                            cmp byte
                                      [ip_buffer+ecx],10
     inc byte[ocount]
     jmp after_oe
                                            je after_scan_esi
even:
     inc byte[ecount]
                                            mov bl,10
                                            mul bl
after_oe:
                                            ;al=al*10
     inc ecx
                                            mov bl,[ip_buffer+ecx]
     jmp repeat_oecount
                                            sub b1,'0'
after_oecount:
                                                            ;b1=
     ret
                                      digit
                                            add al,bl
                                                            ;al=al+bl
                                            inc ecx
print_bl:
                                            imp scan_esi_loop
;; bl should contain value to
                                      after_scan_esi:
                                            mov [esi],al
be printed
     mov ecx,3
     mov esi,2
                                            ret
     mov al,bl
```

FARMAGUDI, PONDA GOA

OUTPUT

```
Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 8C.asm
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ 1d -m elf_i386 -s -o 8C 8C.o
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./8C
ENTER SIZE OF N: 5
ARRAY: 1
3
4
5
EVEN: 002
ODD : 003lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ _
4)
section .data
                                       %macro SYS_READ 2
p1 db 'ENTER SIZE OF N: '
                                            mov eax,3
p1L equ $-p1
                                            mov ebx,2
                                             mov ecx,%1
p2 db 'ENTER ELMENTS INTO YOUR
                                             mov edx,%2
ARRAY:
                                             int 80h
                                       %endmacro
p2L equ $-p2
p3 db 'ARRAY: '
                                       section .text
p3L equ $-p3
                                       global _start
     space db ' '
     pres db "Above 50 : "
                                       _start:
     prl equ $-pres
nres db "Below 50 : "
                                             SYS_WRITE p1,p1L
                                             SYS_READ size,2
     nrl equ $-nres
                                             SYS_WRITE p3,p3L
     newline db 10
                                             sub [size],dword '0'
section .bss
                                             and [size], dword 000fh
     arr resb 10
                                             mov eax, arr
     acount resb 1
                                            mov edx,[size]
     bcount resb 1
                                             call input
     dis_buffer resb 3
     ip_buffer resb 4
                                            mov eax, arr
     size resb 4
                                             mov edx,[size]
                                             call abcount
%macro SYS WRITE 2
     mov eax,4
                                             SYS_WRITE pres,prl
                                             mov bl,[acount]
     mov ebx, 1
                                             call print_bl
     mov ecx,%1
     mov edx,%2
     int 80h
                                             SYS_WRITE newline,1
%endmacro
                                             SYS_WRITE nres,nrl
```

FARMAGUDI, PONDA GOA

```
mov bl,[bcount]
                                            popad
     call print_bl
                                            inc ecx
                                            jmp repeat_input
                                       after_input:
     mov eax,1
     mov ebx,0
     int 80h
                                            ret
abcount:
                                       print_bl:
                                       ;; bl should contain value to
;; eax should contain address
                                       be printed
of array
;; edx should contain size of
                                            mov ecx, 3
                                            mov esi,2
array
;; stores values in acount and
                                            mov al,bl
                                       print_bl_loop:
bcount variables
     mov [acount], byte 0
                                            mov ah,0
     mov [bcount], byte 0
                                            mov dl,10
                                            div dl
     mov ecx,0
                                            add ah, '0'
repeat_abcount:
     cmp ecx,edx
                                            mov [dis_buffer+esi],ah
     jge after_abcount
                                            dec esi
                                            loop print_bl_loop
     mov b1,[eax+ecx]
                                            mov eax,4
     cmp b1,50
                                            mov ebx,1
     jl below
                                            mov ecx,dis_buffer
     jg above
                                            mov edx,3
     jmp after_ba
                                            int 80h
below:
     inc byte[bcount]
                                            ret
     jmp after_ba
above:
                                       scan_esi:
     inc byte[acount]
                                       ;; esi should contain address
after_ba:
                                       of value to be scanned
      inc ecx
     jmp repeat_abcount
                                            mov eax,3
after_abcount:
                                            mov ebx,2
                                            mov ecx, ip_buffer
     ret
                                            mov edx,4
                                            int 80h
input:
                                            mov al,0
;; eax should contain address
                                            mov ecx,0
                                       scan_esi_loop:
of array
;; edx should contain size of
                                            cmp ecx, dword 3
                                            jge after_scan_esi
array
;; scans bytes
     mov ecx.0
                                            cmp byte
                                       [ip_buffer+ecx],10
repeat_input:
     cmp ecx,edx
                                            je after_scan_esi
     jge after_input
                                            mov bl,10
prom:
                                            mul bl
     mov ebx,eax
                                            ;al=al*10
     add ebx,ecx
                                            mov b],[ip_buffer+ecx]
     mov esi,ebx
     pushad
                                            sub b1,'0'
                                                              ;b1=
                                       digit
     call scan_esi
```

FARMAGUDI, PONDA GOA

OUTPUT

```
Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop
```

```
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 8D.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 8D 8D.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./8D
ENTER SIZE OF N: 5
ARRAY: 52
40
30
59
78
Above 50 : 003
Below 50 : 0021loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```

FARMAGUDI, PONDA GOA

```
1)
section .data
                                           mov eax,[num]
msg1 db 'ENTER NUMBER OF
                                           mov [i],eax
ELEMENTS:
msg1len equ $-msg1
                                           mov esi, array
msg2 db 'ENTER THE ELEMENTS
                                           mov eax, num
INTO THE ARRAY: ',10
msg2len equ $-msg2
msg3 db 'ENTER THE ELEMENT TO
                                           inputelement:
                                               mov eax,3
BE SEARCHED:
                                               mov ebx,2
msg3len equ $-msg3
                                               mov ecx, element
                                               mov edx,2
msg4 db 'ELEMENT FOUND AT INDEX
                                               int 80h
msg4len equ $-msg4
msg5 db 'ELEMENT NOT FOUND',10
                                               mov ebx,[element]
msq5len equ $-msg5
                                               mov [esi],ebx
newline db ' ',10
newlinelen equ $-newline
                                               dec byte[i]
                                               inc esi
;array declaration and
initialization
                                               cmp byte[i],'0'
array dw 0,0,0,0,0,0,0,0,0
                                               jne inputelement
arraylen equ 9 ; static
array count
                                           write_message msg3,msg3len
                                           read_message n,9
section .bss
num resb 9
                                           mov eax, [num]
element resb 9
                                           mov [i],eax
i resb 9
n resb 9
                                           mov esi, array
index resb 9
                                           mov eax,i
%macro write_message 2
                                           linearSearch:
    mov eax,4
                                               mov eax,[esi]
    mov ebx,1
                                               mov [element],eax
    mov ecx,%1
    mov edx,%2
                                               mov al,[n]
                                               sub al,'0'
    int 80h
                                               mov bl,[element]
%endmacro
                                               sub b1,'0'
%macro read_message 2
                                               cmp al,bl
    mov eax,3
                                               je exit
    mov ebx,2
    mov ecx,%1
                                               dec byte[i]
    mov edx,%2
                                               inc esi
    int 80h
%endmacro
                                               cmp byte[i],'0'
                                               jne linearSearch
section .text
global _start
                                           write_message msq5,msq5len
                                           mov eax,1
_start:
    write_message msg1,msg1len
                                           mov ebx,0
                                           int 80h
    read_message num,9
    write_message msg2,msg2len
                                           exit:
```

FARMAGUDI, PONDA GOA

```
write_message
msg4,msg4len

mov eax,[num]
sub eax,'0'
mov ebx,[i]
sub ebx,'0'
sub eax,ebx
add eax,'0'
mov [index],eax
mov [index],eax
mov elline,message
newline,newlinelen
mov eax,1
mov eax,1
sub eax,ebx
int 80h
```

OUTPUT

initialization

array count

array dw 0,0,0,0,0,0,0,0,0

arraylen equ 9 ; static

```
Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 9A.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 9A 9A.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./9A
ENTER NUMBER OF ELEMENTS: 5
ENTER THE ELEMENTS INTO THE ARRAY:
2
3
4
ENTER THE ELEMENT TO BE SEARCHED: 3
ELEMENT FOUND AT INDEX 2
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
2)
                                       section .bss
section .data
msg1 db 'ENTER THE ELEMENTS
                                       num resb 9
INTO THE ARRAY: ',10
                                      element resb 9
msgllen equ $-msgl
                                      ele resb 9
msg3 db 'ENTER NUMBER OF
                                       i resb 9
                                       1 resb 9
ELEMENTS:
                                      h resb 9
msg3len equ $-msg3
msg4 db 'ELEMENT FOUND AT
                                      mid resb 9
                                       count resb 9
INDEX:
msq4len equ $-msq4
msg5 db 'ELEMENT FOUND AT: '
                                       %macro calc_mid 3
msg5len equ $-msg5
                                           mov eax,4
msg6 db 'ENTER THE ELEMENT TO
                                           mov ebx,1
                                           mov ecx,''
BE SEARCHED: '
msg6len equ $-msg6
newline db ' ',10
                                           mov edx,0
                                           int 80h
newlinelen equ \u00e4-newline
                                           mov al,[%1]
;array declaration and
                                           sub al,'0'
```

201105025 PAGE NO:

mov bl,[%2]

sub bl,'0' add al,bl

mov bl,'2' sub bl,'0'

FARMAGUDI, PONDA GOA

```
div bl
    add ax, '0'
                                           mov eax,[count]
                                           mov [mid], eax
    mov [%3],ax
%endmacro
                                           mov esi, array
%macro write_message 2
                                           mov eax,i
    mov eax,4
                                           mov esi, array
    mov ebx, 1
    mov ecx,%1
                                           mov eax,i
    mov edx,%2
                                           jmp search
    int 80h
%endmacro
                                           inc_1:
                                               inc byte[count]
%macro read_message 2
                                               dec byte[mid]
                                               call float_int
    mov eax, 3
    mov ebx,2
                                           float_int:
    mov ecx,%1
                                               cmp byte[mid],'1'
    mov edx,%2
    int 80h
                                               jge inc_1
%endmacro
                                               ret
section .text
                                           incr:
global _start
                                               inc esi
                                               inc bl
                                               jmp array_index
_start:
    write_message msg3,msg3len
    read_message num,9
                                           array_index:
                                               cmp al,bl
    write_message msg1,msg1len
                                               jge incr
    mov eax,[num]
                                               ret
    mov [i],eax
                                           search:
                                               mov al,[1]
    mov esi, array
    mov eax, num
                                               mov bl,[h]
                                               cmp al,bl
    inputelement:
                                               jg not_found
        read_message element,2
                                               calc_mid 1,h,mid
                                               mov al,'0'
        mov ebx,[element]
        mov [esi],ebx
                                               mov [count],al
                                               call float_int
        dec byte[num]
        inc esi
                                               mov esi, array
                                               mov eax,[i]
        cmp byte[num],'0'
        ine inputelement
                                               mov al, [count]
                                               mov bl,'1'
    write_message msg6,msg6len
                                               call array_index
    read_message ele,9
                                               mov eax, [esi]
    mov ebx, '0'
                                               mov [element],eax
    mov [1],ebx
                                               mov al,[ele]
    mov eax,[i]
    mov [h],eax
                                               mov bl,[element]
    dec byte[h]
                                               cmp al,bl
    calc_mid 1,h,mid
                                               il left
```

FARMAGUDI, PONDA GOA

```
jg right
je found
                                             found:
                                                  ;inc byte[count]
left:
    mov al,[count]
                                                  write_message
    sub al,'0'
mov bl,'1'
sub bl,'0'
                                   msg4,msg4len
                                                  write_message
                                   count,9
     sub al,bl
                                                  jmp exit
     add al,'0'
     mov [h],al
                                             not_found:
     jmp search
                                                  write_message
                                   msg5,msg5len
right:
     mov al,[count]
                                        exit:
    sub al,'0'
mov bl,'1'
sub bl,'0'
                                             write_message
                                   newline, newlinelen
     add al,bl
                                             mov eax,1
     add al,'0'
                                             mov ebx,0
                                             int 80h
     mov [1],al
     imp search
```

OUTPUT

Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop

```
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 9B.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 9B 9B.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./9B
ENTER NUMBER OF ELEMENTS:5
ENTER THE ELEMENTS INTO THE ARRAY:
1
2
3
4
5
ENTER THE ELEMENT TO BE SEARCHED:4
ELEMENT FOUND AT INDEX: 3
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```

FARMAGUDI, PONDA GOA

```
1)
section .data
                                              mov [j],eax
     mss1 db "ENTER NUMBER OF
                                              call bubble_sort
ELEMENTS: '
                                              write mss3,p3len
     plen equ $-mss1
                                              call display
     mss2 db "ENTER THE
ELEMENTS INTO YOUR ARRAY:",10
                                              mov eax,1
     p2len equ $-mss2
mss3 db "SORTED
                                              mov ebx,0
                                              int 80h
ARRAY:",10
     p3len equ $-mss3
mss4 db "PASS"
                                        input:
                                              write mss2,p2len
     p4len equ $-mss4
                                              mov [i], dword '0'
     mss5 db " -> "
                                        loop1:
     p5len equ $-mss5
                                              mov esi,[i]
     newline db 10 space db ' '
                                              cmp esi,[n]
                                              jge after1
section .bss
                                              sub esi,'0'
                                              add esi,arr
     n resb 4
     arr resb 10
                                              read esi,1
     i resb 4
      j resb 9
                                              inc dword[i]
     trash resb 1
                                              jmp loop1
                                        after1:
%macro write 2
                                              ret
     mov eax,4
     mov ebx,1
     mov ecx,%1
                                        display:
     mov edx,%2
                                              write mss4,p4len
     int 80h
                                              write j,9
                                              write mss5,p5len
%endmacro
                                              mov [i], dword '0'
%macro read 2
                                        loop2:
     mov eax,3
                                              mov esi,[i]
     mov ebx,2
                                              cmp esi,[n]
     mov ecx,%1
                                              jge after2
     mov edx,%2
     int 80h
                                              sub esi,'0'
                                              add esi,arr
     mov eax,3
     mov ebx,2
                                              write esi,1
     mov ecx, trash
                                              write space,1
     mov edx,1
     int 80h
                                              inc dword[i]
%endmacro
                                              jmp loop2
                                        after2:
section .text
                                              write newline,1
     global _start
                                              ret
_start:
     write mss1,plen
                                        bubble_sort:
                                                               ;al is
                                              mov al,0
     read n,1
                                        counter for outer loop,
                                        initialise to 0
     call input
                                              mov bl,[n]
     write newline,1
     mov eax, '0'
                                              sub b1,'0'
```

FARMAGUDI, PONDA GOA

```
sub bl,1
                       ;bl is n-
                                            mov esi,arr
                                            add esi,ecx
1
                                                                    ;esi
loop3:
                                       points to arr[ecx]
     cmp al,bl
                       ;repeat
until al<n-1
                                            mov ah,[esi]
     jge after3
                                            mov bh,[esi+1]
                                            cmp ah,bh
                                                              ;if ah>bh
     pushad
                                            jle after5
     call display
     popad
                                            mov [esi+1],ah
                                            ;swap esi and esi+1
                                            mov [esi],bh
     mov ecx,0
                       ;ecx is
counter for inner loop,
initialise to 0
                                       after5:
     mov dl,bl
                                            inc cl
     sub dl,al
                       ;dl is n-
                                            jmp loop4
                                       after4:
1-al
loop4:
                                            inc al
     cmp cl,dl
                                            inc byte[j]
                       ;repeat
until cl<n-1-al
                                            jmp loop3
                                       after3:
     jge after4
                                            ret
```

OUTPUT

```
Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop
```

```
1loyd@LLOYD:~$ cd /mnt/c/Users/lloyd/Desktop
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 10A.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 10A 10A.o
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./10A
ENTER NUMBER OF ELEMENTS: 5
ENTER THE ELEMENTS INTO YOUR ARRAY:
6
1
3
2
PASS 0 -> 5 6 1 3 2
PASS 1 -> 5 1 3 2 6
PASS 2 -> 1 3 2 5 6
PASS 3 -> 1 2 3 5 6
SORTED ARRAY:
PASS 4 -> 1 2 3 5 6
1loyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ _
```

FARMAGUDI, PONDA GOA

```
2)
section .data
                                              call input
     mss1 db "ENTER NUMBER OF
                                              call insertion_sort
ELEMENTS: '
                                              write mss3,p3len
     plen equ $-mss1
                                              call display
     mss2 db "ENTER THE
ELEMENTS INTO YOUR ARRAY:",10
                                              mov eax,1
     p2len equ $-mss2
mss3 db "SORTED
                                              mov ebx,0
                                              int 80h
ARRAY:",10
     p3len equ $-mss3
mss4 db "PASS"
                                        input:
                                              write mss2,p2len
     p4len equ $-mss4
                                              mov [i], dword '0'
     mss5 db " -> "
                                        loop1:
     p5len equ $-mss5
                                              mov esi,[i]
     newline db 10 space db ' '
                                              cmp esi,[n]
                                              jge after1
section .bss
                                              sub esi,'0'
                                              add esi,arr
     n resb 4
     arr resb 10
                                              read esi,1
     i resb 4
      j resb 9
                                              inc dword[i]
     trash resb 1
                                              jmp loop1
                                        after1:
%macro write 2
                                              ret
     mov eax,4
     mov ebx,1
     mov ecx,%1
                                        display:
     mov edx,%2
                                              write mss4,p4len
     int 80h
                                              write j,9
                                              write mss5,p5len
%endmacro
                                              mov [i], dword '0'
%macro read 2
                                        loop2:
     mov eax,3
                                              mov esi,[i]
     mov ebx,2
                                              cmp esi,[n]
     mov ecx,%1
                                              jge after2
     mov edx,%2
     int 80h
                                              sub esi,'0'
                                              add esi,arr
     mov eax,3
     mov ebx,2
                                              write esi,1
     mov ecx, trash
                                              write space,1
     mov edx,1
     int 80h
                                              inc dword[i]
%endmacro
                                              jmp loop2
                                        after2:
section .text
                                              write newline,1
     global _start
                                              ret
_start:
     write mss1,plen
                                        insertion_sort:
                                                                ;al is
                                              mov eax,1
     read n,1
                                        counter for outer loop,
                                        initialise to 1
     mov eax, '0'
     mov [j],eax
                                              mov b1,[n]
                                                                ;b1=n
                                              sub b1,'0'
```

FARMAGUDI, PONDA GOA

```
loop3:
                                            cmp dl,[arr+ecx]
     cmp al,bl
                       ;repeat
                                            ;...dl<arr[cl]
until al<n
                                            jge after4
     jge after3
                                            mov dh,[arr+ecx]
                                            mov [arr+ecx+1],dh
     pushad
     call display
                                            ;arr[ecx], arr[ecx+1]
     popad
                                            dec ecx
                                            ;ecx--
     mov ecx,0
     mov cl,al
                                            jmp loop4
                       ;cl is
                                      after4:
     sub cl,1
counter for inner loop,
                                            mov [arr+ecx+1],d]
initialise to al-1
                                            ;arr[ecx+1]=dl
     mov dl,[arr+eax]
                                            inc al
     ;store arr[eax] in dl
                                            ;al++
loop4:
                                            inc byte[j]
                                            jmp loop3
     cmp c1,0
                       ;repeat
until c1>=0 and...
                                      after3:
     jl after4
                                            ret
```

OUTPUT

Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop

```
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 10B.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 10B 10B.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./10B
ENTER NUMBER OF ELEMENTS: 5
ENTER THE ELEMENTS INTO YOUR ARRAY:
5
6
1
3
2
PASS 0 -> 5 6 1 3 2
PASS 1 -> 5 6 1 3 2
PASS 2 -> 1 5 6 3 2
PASS 3 -> 1 3 5 6 2
SORTED ARRAY:
PASS 4 -> 1 2 3 5 6
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ __
```

FARMAGUDI, PONDA GOA

```
3)
section .data
     mss1 db "ENTER NUMBER OF
                                              call input
ELEMENTS: '
                                              call selection_sort
     plen equ $-mss1
                                              write mss3,p3len
     mss2 db "ENTER THE
                                              call display
ELEMENTS INTO YOUR ARRAY:",10
     p2len equ $-mss2
mss3 db "SORTED
                                              mov eax,1
                                              mov ebx,0
ARRAY:",10
                                              int 80h
     p3len equ $-mss3
mss4 db "PASS"
                                        input:
     p4len equ $-mss4
                                              write mss2,p2len
     mss5 db " -> "
                                              mov [i],dword 0
     p5len equ $-mss5
                                        loop1:
     newline db 10 space db ' '
                                              mov esi,[i]
                                              cmp esi,[n]
                                              jge after1
section .bss
                                              add esi,arr
     n resb 4
     arr resb 10
                                              read esi,1
     i resb 4
      j resb 9
                                              inc dword[i]
     trash resb 1
                                              jmp loop1
                                        after1:
%macro write 2
                                              ret
     mov eax,4
     mov ebx,1
     mov ecx,%1
                                        display:
     mov edx,%2
                                              write mss4,p4len
     int 80h
                                              write j,9
                                              write mss5,p5len
%endmacro
                                              mov [i], dword 0
%macro read 2
                                        loop2:
     mov eax,3
                                              mov esi,[i]
     mov ebx,2
                                              cmp esi,[n]
     mov ecx,%1
                                              jge after2
     mov edx,%2
     int 80h
                                              add esi,arr
     mov eax,3
                                              write esi,1
     mov ebx,2
                                              write space,1
     mov ecx, trash
     mov edx,1
                                              inc dword[i]
     int 80h
                                              jmp loop2
%endmacro
                                        after2:
                                              write newline,1
section .text
                                              ret
     global _start
                                        selection_sort:
_start:
                                                                ;al is
     write mss1,plen
                                              mov eax,0
                                        counter for outer loop,
     read n,1
     sub byte[n],'0'
                                        initialise to 0
                                              mov bl,[n]
     mov eax, '0'
                                                               ;bl is n-
                                              sub bl,1
     mov [j],eax
```

FARMAGUDI, PONDA GOA

```
loop3:
                                            mov bh,[esi]
     cmp al,bl
                       ;repeat
until al<n-1
                                            mov dh,[edi]
     jge after3
                                            cmp bh,dh
                                                             ;if bh<dh
                                            jge after5
     pushad
     call display
                                            mov edi,arr
                                                                   ;edi
                                      points to arr[ecx]
     popad
                                            add edi,ecx
     mov ecx,0
                      ;clear
                                      after5:
ecx register
                                            inc cl
                                            jmp loop4
     mov cl,al
                       ;ecx is
counter for inner loop,
                                      after4:
                                            ;; swap arr[eax] and
initialise to al+1
     add cl,1
                                       [edi]
                                            mov bh,[arr+eax]
                                            mov dh,[edi]
     mov edi,arr
     add edi,eax
                            ;edi
                                            mov [arr+eax],dh
                                            mov [edi],bh
points to arr[eax]
loop4:
     cmp cl,[n]
                      ;repeat
                                            inc al
until cl<n
                                            inc byte[j]
     jge after4
                                            jmp loop3
                                      after3:
     mov esi,arr
                                            ret
     add esi,ecx
                            ;esi
points to arr[ecx]
```

OUTPUT

```
Iloyd@LLOYD: /mnt/c/Users/Iloyd/Desktop
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 10C.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 10C 10C.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./10C
ENTER NUMBER OF ELEMENTS: 5
ENTER THE ELEMENTS INTO YOUR ARRAY:
6
1
3
PASS 0 -> 5 6 1 3 2
PASS 1 -> 1 6 5 3 2
PASS 2 -> 1 2 5 3 6
PASS 3 -> 1 2 3 5 6
SORTED ARRAY:
PASS 4 -> 1 2 3 5 6
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```