1)

```
%macro print 2
      mov eax,sys_out
      mov ebx,stdout
      mov ecx,%1
      mov edx,%2
      int 80h
      %endmacro

      %macro input 2
      mov eax,sys_in
      mov ebx,stdin
      mov ecx,%1
      mov edx,%2
      int 80h
      %endmacro

      section .data
      sys_out equ 4 ;To output
      sys_in equ 3 ;To Input

      stdout equ 1 ;Stdout
      stdin equ 2 ;Stdins

      p1 db 'ENTER SIZE OF N: '
      p1L equ $-p1

      p2 db 'ENTER ELMENTS INTO
YOUR ARRAY: '
      p2L equ $-p2

      p3 db 'ARRAY: '
      p3L equ $-p3

      space db ' '
      spaceL equ $-space

      newline db ' '
      newlineL equ $-newline


      section .bss
      size resb 9
      arr resb 10
      dis_buffer resb 3
      ip_buffer resb 4


section _text:
global _start:
  _start:
  print p1,p1L
  input size,9
  print p2,p2L

  sub [size],dword '0'
  and [size],dword 000fh
  mov eax,arr
  mov edx,[size]
  call inputElArr

  print p3,p3L
  mov eax,arr
  mov edx,[size]
  call display

  mov eax,1
  mov ebx,0
  int 80h



  display:
  ;; eax should contain address
of array
  ;; edx should contain size of
array
  ;; prints bytes
  mov ecx,0
  repeat_display:
  cmp ecx,edx
  jge after_display

  mov bl,[eax+ecx]
  pushad
  call print_bl

  mov eax,4
  mov ebx,1
  mov ecx,space
  mov edx,spaceL
  int 80h

  popad
  inc ecx
  jmp repeat_display
  after_display:
  mov eax,4
  mov ebx,1
  mov ecx,newline
  mov edx,newlineL
  int 80h

  ret


inputElArr:
  ;; eax should contain address
of array
  ;; edx should contain size of
array
```

```
  ;; scans bytes
  mov ecx,0
  repeat_input:
  cmp ecx,edx
  jge after_input

  mov ebx,eax
  add ebx,ecx
  mov esi,ebx
  pushad
  call scan_esi
  popad
  inc ecx
  jmp repeat_input
  after_input:

  ret

  print_bl:
  ;; bl should contain value to
be printed
  mov ecx,3
  mov esi,2
  mov al,bl
  print_bl_loop:
  mov ah,0
  mov dl,10
  div dl
  add ah,'0'
  mov [dis_buffer+esi],ah
  dec esi
  loop print_bl_loop

  mov eax,4
  mov ebx,1
  mov ecx,dis_buffer
  mov edx,3
  int 80h

  ret

  scan_esi:
  ;; esi should contain address
of value to be scanned
  mov eax,3
  mov ebx,2
  mov ecx,ip_buffer
  mov edx,4
  int 80h

  mov al,0
  mov ecx,0
  scan_esi_loop:
  cmp ecx,dword 3
  jge after_scan_esi

  cmp byte [ip_buffer+ecx],10
  je after_scan_esi

  mov bl,10
  mul bl              ;al=al*10
  mov bl,[ip_buffer+ecx]
  sub bl,'0'          ;bl=
digit
  add al,bl           ;al=al+bl

  inc ecx
  jmp scan_esi_loop

  after_scan_esi:
  mov [esi],al

  ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/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$
```

2)

```
section .data
pres db "POSITIVE NUMBER COUNT:
"
prl equ $-pres
nres db "NEGATIVE NUMBER COUNT:
"
nrl equ $-nres
arr db 4,-10,-15,18,28,26,30,
-27,1,-2
newline db 10

section .bss
pcount resb 1
ncount resb 1
dis_buffer resb 3

%macro SYS_WRITE 2
mov eax,4
mov ebx,1
mov ecx,%1
mov edx,%2
int 80h
%endmacro

section .text
global _start

_start:
mov eax,arr
mov edx,10
call pncount

SYS_WRITE pres,prl
mov bl,[pcount]
call print_bl

SYS_WRITE newline,1

SYS_WRITE nres,nrl
mov bl,[ncount]
call print_bl

mov eax,1
mov ebx,0
int 80h


pncount:
;; eax should contain address
of array
;; edx should contain size of
array
;; stores values in pcount and
ncount variables
mov [pcount],byte 0
mov [ncount],byte 0
mov ecx,0
repeat_pncount:
cmp ecx,edx
jge after_pncount

mov bl,[eax+ecx]

cmp bl,0
jl negative
jg positive
jmp after_np
negative:
inc byte[ncount]
jmp after_np
positive:
inc byte[pcount]
after_np:
inc ecx
jmp repeat_pncount
after_pncount:
ret



print_bl:
;; bl should contain value to
be printed
mov ecx,3
mov esi,2
mov al,bl
print_bl_loop:
mov ah,0
mov dl,10
div dl
add ah,'0'
mov [dis_buffer+esi],ah
dec esi
loop print_bl_loop

mov eax,4
mov ebx,1
mov ecx,dis_buffer
mov edx,3
int 80h

ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/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
lloyd@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: '
  p1L equ $-p1

  p2 db 'ENTER ELMENTS INTO
YOUR ARRAY: '
  p2L equ $-p2

  p3 db 'ARRAY: '
  p3L equ $-p3
      space db ' '
      ores db "ODD : "
      orl equ $-ores
      eres db "EVEN : "
      erl equ $-eres
      newline db 10

section .bss
      arr resb 10
      ocount resb 1
      ecount resb 1
      dis_buffer resb 3
      ip_buffer resb 4
      size resb 4

%macro SYS_WRITE 2
      mov eax,4
      mov ebx,1
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro

%macro SYS_READ 2
      mov eax,3
      mov ebx,2
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro


section .text
global _start
```

```
_start:
      SYS_WRITE p1,p1L
      SYS_READ size,2
      SYS_WRITE p3,p3L

      sub [size],dword '0'
      and [size],dword 000fh
      mov eax,arr
      mov edx,[size]
      call input

      mov esi,arr
      mov edx,[size]
      call oecount

      SYS_WRITE eres,erl
      mov bl,[ecount]
      call print_bl

      SYS_WRITE newline,1

      SYS_WRITE ores,orl
      mov bl,[ocount]
      call print_bl

      mov eax,1
      mov ebx,0
      int 80h


input:
      ;; eax should contain
address of array
      ;; edx should contain
size of array
      ;; scans bytes
      mov ecx,0
repeat_input:
      cmp ecx,edx
      jge after_input

      mov ebx,eax
      add ebx,ecx
```

```
      mov esi,ebx
      pushad
      call scan_esi
      popad
      inc ecx
      jmp repeat_input
after_input:
      ret



oecount:
;; esi should contain address
of array
;; edx should contain size of
array
;; stores values in ocount and
ecount variables
      mov [ocount],byte 0
      mov [ecount],byte 0
      mov ecx,0
repeat_oecount:
      cmp ecx,edx
      jge after_oecount

      mov al,[esi+ecx]
      cbw
      mov bl,2
      div bl
      ;al=al/2 , ah=al%2
      cmp ah,0
      je even               ;if
(ah%2)==0
odd:
      inc byte[ocount]
      jmp after_oe
even:
      inc byte[ecount]
after_oe:
      inc ecx
      jmp repeat_oecount
after_oecount:
      ret



print_bl:
;; bl should contain value to
be printed
      mov ecx,3
      mov esi,2
      mov al,bl
print_bl_loop:
      mov ah,0
      mov dl,10
      div dl
      add ah,'0'
      mov [dis_buffer+esi],ah
      dec esi
      loop print_bl_loop

      mov eax,4
      mov ebx,1
      mov ecx,dis_buffer
      mov edx,3
      int 80h

      ret

scan_esi:
      ;; esi should contain
address of value to be scanned
      mov eax,3
      mov ebx,2
      mov ecx,ip_buffer
      mov edx,4
      int 80h

      mov al,0
      mov ecx,0

scan_esi_loop:
      cmp ecx,dword 3
      jge after_scan_esi

      cmp byte
[ip_buffer+ecx],10
      je after_scan_esi

      mov bl,10
      mul bl
      ;al=al*10
      mov bl,[ip_buffer+ecx]
      sub bl,'0'       ;bl=
digit
      add al,bl        ;al=al+bl

      inc ecx
      jmp scan_esi_loop
after_scan_esi:
      mov [esi],al

      ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/Desktop
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ nasm -f elf 8C.asm
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ld -m elf_i386 -s -o 8C 8C.o
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./8C
ENTER SIZE OF N: 5
ARRAY: 1
2
3
4
5
EVEN : 002
ODD : 003lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```

4)

```
section .data
p1 db 'ENTER SIZE OF N: '
p1L equ $-p1

p2 db 'ENTER ELMENTS INTO YOUR
ARRAY: '
p2L equ $-p2

p3 db 'ARRAY: '
p3L equ $-p3
      space db ' '
      pres db "Above 50 : "
      prl equ $-pres
      nres db "Below 50 : "
      nrl equ $-nres
      newline db 10

section .bss
      arr resb 10
      acount resb 1
      bcount resb 1
      dis_buffer resb 3
      ip_buffer resb 4
      size resb 4

%macro SYS_WRITE 2
      mov eax,4
      mov ebx,1
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro

%macro SYS_READ 2
      mov eax,3
      mov ebx,2
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro

section .text
global _start

_start:
      SYS_WRITE p1,p1L
      SYS_READ size,2
      SYS_WRITE p3,p3L

      sub [size],dword '0'
      and [size],dword 000fh
      mov eax,arr
      mov edx,[size]
      call input

      mov eax,arr
      mov edx,[size]
      call abcount

      SYS_WRITE pres,prl
      mov bl,[acount]
      call print_bl

      SYS_WRITE newline,1

      SYS_WRITE nres,nrl
```

```
      mov bl,[bcount]
      call print_bl

      mov eax,1
      mov ebx,0
      int 80h


abcount:
;; eax should contain address
of array
;; edx should contain size of
array
;; stores values in acount and
bcount variables
      mov [acount],byte 0
      mov [bcount],byte 0
      mov ecx,0
repeat_abcount:
      cmp ecx,edx
      jge after_abcount

      mov bl,[eax+ecx]

      cmp bl,50
      jl below
      jg above
      jmp after_ba
below:
      inc byte[bcount]
      jmp after_ba
above:
      inc byte[acount]
after_ba:
      inc ecx
      jmp repeat_abcount
after_abcount:
      ret



input:
;; eax should contain address
of array
;; edx should contain size of
array
;; scans bytes
      mov ecx,0
repeat_input:
      cmp ecx,edx
      jge after_input
prom:
      mov ebx,eax
      add ebx,ecx
      mov esi,ebx
      pushad
      call scan_esi
      popad
      inc ecx
      jmp repeat_input
after_input:

      ret


print_bl:
;; bl should contain value to
be printed
      mov ecx,3
      mov esi,2
      mov al,bl
print_bl_loop:
      mov ah,0
      mov dl,10
      div dl
      add ah,'0'
      mov [dis_buffer+esi],ah
      dec esi
      loop print_bl_loop

      mov eax,4
      mov ebx,1
      mov ecx,dis_buffer
      mov edx,3
      int 80h

      ret


scan_esi:
;; esi should contain address
of value to be scanned
      mov eax,3
      mov ebx,2
      mov ecx,ip_buffer
      mov edx,4
      int 80h

      mov al,0
      mov ecx,0
scan_esi_loop:
      cmp ecx,dword 3
      jge after_scan_esi

      cmp byte
[ip_buffer+ecx],10
      je after_scan_esi

      mov bl,10
      mul bl
      ;al=al*10
      mov bl,[ip_buffer+ecx]
      sub bl,'0'       ;bl=
digit
```

```
        add al,bl          ;al=al+bl        after_scan_esi:
                                                    mov [esi],al
        inc ecx
        jmp scan_esi_loop                           ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/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 : 002lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```

1)

```
section .data
msg1 db 'ENTER NUMBER OF
ELEMENTS: '
msg1len equ $-msg1
msg2 db 'ENTER THE ELEMENTS
INTO THE ARRAY:',10
msg2len equ $-msg2
msg3 db 'ENTER THE ELEMENT TO
BE SEARCHED:'
msg3len equ $-msg3
msg4 db 'ELEMENT FOUND AT INDEX
'
msg4len equ $-msg4
msg5 db 'ELEMENT NOT FOUND',10
msg5len equ $-msg5
newline db '    ',10
newlinelen equ $-newline

;array declaration and
initialization
array dw 0,0,0,0,0,0,0,0,0
arraylen equ 9    ; static
array count

section .bss
num resb 9
element resb 9
i resb 9
n resb 9
index resb 9

%macro write_message 2
    mov eax,4
    mov ebx,1
    mov ecx,%1
    mov edx,%2
    int 80h
%endmacro

%macro read_message 2
    mov eax,3
    mov ebx,2
    mov ecx,%1
    mov edx,%2
    int 80h
%endmacro

section .text
global _start

_start:
    write_message msg1,msg1len
    read_message num,9
    write_message msg2,msg2len
    mov eax,[num]
    mov [i],eax

    mov esi,array
    mov eax,num

inputelement:
    mov eax,3
    mov ebx,2
    mov ecx,element
    mov edx,2
    int 80h

    mov ebx,[element]
    mov [esi],ebx

    dec byte[i]
    inc esi

    cmp byte[i],'0'
    jne inputelement

write_message msg3,msg3len
read_message n,9

mov eax,[num]
mov [i],eax

mov esi,array
mov eax,i

linearSearch:
    mov eax,[esi]
    mov [element],eax

    mov al,[n]
    sub al,'0'
    mov bl,[element]
    sub bl,'0'
    cmp al,bl
    je exit

    dec byte[i]
    inc esi

    cmp byte[i],'0'
    jne linearSearch

write_message msg5,msg5len

mov eax,1
mov ebx,0
int 80h

exit:
```

```
        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

        write_message index,9
        write_message
newline,newlinelen

        mov eax,1
        mov ebx,0
        int 80h
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/Desktop
lloyd@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:
1
2
3
4
5
ENTER THE ELEMENT TO BE SEARCHED:3
ELEMENT FOUND AT INDEX 2
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```

2)

```
section .data
msg1 db 'ENTER THE ELEMENTS
INTO THE ARRAY:',10
msg1len equ $-msg1
msg3 db 'ENTER NUMBER OF
ELEMENTS:'
msg3len equ $-msg3
msg4 db 'ELEMENT FOUND AT
INDEX:  '
msg4len equ $-msg4
msg5 db 'ELEMENT FOUND AT: '
msg5len equ $-msg5
msg6 db 'ENTER THE ELEMENT TO
BE SEARCHED:'
msg6len equ $-msg6
newline db '    ',10
newlinelen equ $-newline

;array declaration and
initialization
array dw 0,0,0,0,0,0,0,0,0
arraylen equ 9    ; static
array count
```

```
section .bss
num resb 9
element resb 9
ele resb 9
i resb 9
l resb 9
h resb 9
mid resb 9
count resb 9

%macro calc_mid 3
    mov eax,4
    mov ebx,1
    mov ecx,''
    mov edx,0
    int 80h

    mov al,[%1]
    sub al,'0'
    mov bl,[%2]
    sub bl,'0'
    add al,bl
    mov bl,'2'
    sub bl,'0'
```

```
    div bl
    add ax,'0'
    mov [%3],ax
%endmacro

%macro write_message 2
    mov eax,4
    mov ebx,1
    mov ecx,%1
    mov edx,%2
    int 80h
%endmacro

%macro read_message 2
    mov eax,3
    mov ebx,2
    mov ecx,%1
    mov edx,%2
    int 80h
%endmacro

section .text
global _start

_start:
    write_message msg3,msg3len
    read_message num,9
    write_message msg1,msg1len

    mov eax,[num]
    mov [i],eax

    mov esi,array
    mov eax,num

    inputelement:
        read_message element,2

        mov ebx,[element]
        mov [esi],ebx

        dec byte[num]
        inc esi

        cmp byte[num],'0'
        jne inputelement

    write_message msg6,msg6len
    read_message ele,9

    mov ebx,'0'
    mov [l],ebx
    mov eax,[i]
    mov [h],eax
    dec byte[h]

    calc_mid l,h,mid

    mov eax,[count]
    mov [mid],eax

    mov esi,array
    mov eax,i

    mov esi,array
    mov eax,i
    jmp search

    inc_1:
        inc byte[count]
        dec byte[mid]
        call float_int

    float_int:
        cmp byte[mid],'1'
        jge inc_1
        ret

    incr:
        inc esi
        inc bl
        jmp array_index

    array_index:
        cmp al,bl
        jge incr
        ret

    search:
        mov al,[l]
        mov bl,[h]
        cmp al,bl
        jg not_found

        calc_mid l,h,mid
        mov al,'0'
        mov [count],al
        call float_int

        mov esi,array
        mov eax,[i]

        mov al,[count]
        mov bl,'1'
        call array_index

        mov eax,[esi]
        mov [element],eax

        mov al,[ele]
        mov bl,[element]

        cmp al,bl
        jl left
```

```
        jg right
        je found

        left:
            mov al,[count]
            sub al,'0'
            mov bl,'1'
            sub bl,'0'
            sub al,bl
            add al,'0'
            mov [h],al
            jmp search

        right:
            mov al,[count]
            sub al,'0'
            mov bl,'1'
            sub bl,'0'
            add al,bl
            add al,'0'
            mov [l],al
            jmp search
```

```
        found:
            ;inc byte[count]

            write_message
msg4,msg4len
            write_message
count,9
            jmp exit

        not_found:
            write_message
msg5,msg5len

    exit:
        write_message
newline,newlinelen

        mov eax,1
        mov ebx,0
        int 80h
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/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$
```

1)

```
section .data
      mss1 db "ENTER NUMBER OF
ELEMENTS: "
      plen equ $-mss1
      mss2 db "ENTER THE
ELEMENTS INTO YOUR ARRAY:",10
      p2len equ $-mss2
      mss3 db "SORTED
ARRAY:",10
      p3len equ $-mss3
      mss4 db "PASS "
      p4len equ $-mss4
      mss5 db " -> "
      p5len equ $-mss5
      newline db 10
      space db ' '

section .bss
      n resb 4
      arr resb 10
      i resb 4
      j resb 9
      trash resb 1

%macro write 2
      mov eax,4
      mov ebx,1
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro

%macro read 2
      mov eax,3
      mov ebx,2
      mov ecx,%1
      mov edx,%2
      int 80h
      mov eax,3
      mov ebx,2
      mov ecx,trash
      mov edx,1
      int 80h
%endmacro

section .text
      global _start

_start:
      write mss1,plen
      read n,1

      call input
      write newline,1
      mov eax,'0'
      mov [j],eax
      call bubble_sort
      write mss3,p3len
      call display

      mov eax,1
      mov ebx,0
      int 80h

input:
      write mss2,p2len
      mov [i],dword '0'
loop1:
      mov esi,[i]
      cmp esi,[n]
      jge after1

      sub esi,'0'
      add esi,arr
      read esi,1

      inc dword[i]
      jmp loop1
after1:
      ret


display:
      write mss4,p4len
      write j,9
      write mss5,p5len
      mov [i],dword '0'
loop2:
      mov esi,[i]
      cmp esi,[n]
      jge after2

      sub esi,'0'
      add esi,arr
      write esi,1
      write space,1

      inc dword[i]
      jmp loop2
after2:
      write newline,1
      ret


bubble_sort:
      mov al,0        ;al is
counter for outer loop,
initialise to 0
      mov bl,[n]
      sub bl,'0'
```

```
      sub bl,1          ;bl is n-
1
loop3:
      cmp al,bl         ;repeat
until al<n-1
      jge after3

      pushad
      call display
      popad

      mov ecx,0         ;ecx is
counter for inner loop,
initialise to 0
      mov dl,bl
      sub dl,al         ;dl is n-
1-al
loop4:
      cmp cl,dl         ;repeat
until cl<n-1-al
      jge after4
            mov esi,arr
            add esi,ecx           ;esi
points to arr[ecx]

            mov ah,[esi]
            mov bh,[esi+1]
            cmp ah,bh
            jle after5      ;if ah>bh

            mov [esi+1],ah
            ;swap esi and esi+1
            mov [esi],bh

after5:
            inc cl
            jmp loop4
after4:
            inc al
            inc byte[j]
            jmp loop3
after3:
            ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/Desktop
lloyd@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
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$ ./10A
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 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
lloyd@LLOYD:/mnt/c/Users/lloyd/Desktop$
```

2)

```
section .data
      mss1 db "ENTER NUMBER OF
ELEMENTS: "
      plen equ $-mss1
      mss2 db "ENTER THE
ELEMENTS INTO YOUR ARRAY:",10
      p2len equ $-mss2
      mss3 db "SORTED
ARRAY:",10
      p3len equ $-mss3
      mss4 db "PASS "
      p4len equ $-mss4
      mss5 db " -> "
      p5len equ $-mss5
      newline db 10
      space db ' '

section .bss
      n resb 4
      arr resb 10
      i resb 4
      j resb 9
      trash resb 1

%macro write 2
      mov eax,4
      mov ebx,1
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro

%macro read 2
      mov eax,3
      mov ebx,2
      mov ecx,%1
      mov edx,%2
      int 80h
      mov eax,3
      mov ebx,2
      mov ecx,trash
      mov edx,1
      int 80h
%endmacro

section .text
      global _start

_start:
      write mss1,plen
      read n,1

      mov eax,'0'
      mov [j],eax

      call input
      call insertion_sort
      write mss3,p3len
      call display

      mov eax,1
      mov ebx,0
      int 80h

input:
      write mss2,p2len
      mov [i],dword '0'
loop1:
      mov esi,[i]
      cmp esi,[n]
      jge after1

      sub esi,'0'
      add esi,arr
      read esi,1

      inc dword[i]
      jmp loop1
after1:
      ret


display:
      write mss4,p4len
      write j,9
      write mss5,p5len
      mov [i],dword '0'
loop2:
      mov esi,[i]
      cmp esi,[n]
      jge after2

      sub esi,'0'
      add esi,arr
      write esi,1
      write space,1

      inc dword[i]
      jmp loop2
after2:
      write newline,1
      ret


insertion_sort:
      mov eax,1       ;al is
counter for outer loop,
initialise to 1
      mov bl,[n]      ;bl=n
      sub bl,'0'
```

```
loop3:
      cmp al,bl         ;repeat
until al<n
      jge after3

      pushad
      call display
      popad

      mov ecx,0
      mov cl,al
      sub cl,1          ;cl is
counter for inner loop,
initialise to al-1

      mov dl,[arr+eax]
      ;store arr[eax] in dl
loop4:
      cmp cl,0          ;repeat
until cl>=0 and...
      jl after4
                              cmp dl,[arr+ecx]
                              ;...dl<arr[cl]
                              jge after4

                              mov dh,[arr+ecx]
                              mov [arr+ecx+1],dh
                              ;arr[ecx], arr[ecx+1]

                              dec ecx
                              ;ecx--
                              jmp loop4
                        after4:
                              mov [arr+ecx+1],dl
                              ;arr[ecx+1]=dl

                              inc al
                              ;al++
                              inc byte[j]
                              jmp loop3
                        after3:
                              ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/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$
```

3)

```
section .data
      mss1 db "ENTER NUMBER OF
ELEMENTS: "
      plen equ $-mss1
      mss2 db "ENTER THE
ELEMENTS INTO YOUR ARRAY:",10
      p2len equ $-mss2
      mss3 db "SORTED
ARRAY:",10
      p3len equ $-mss3
      mss4 db "PASS "
      p4len equ $-mss4
      mss5 db " -> "
      p5len equ $-mss5
      newline db 10
      space db ' '

section .bss
      n resb 4
      arr resb 10
      i resb 4
      j resb 9
      trash resb 1

%macro write 2
      mov eax,4
      mov ebx,1
      mov ecx,%1
      mov edx,%2
      int 80h
%endmacro

%macro read 2
      mov eax,3
      mov ebx,2
      mov ecx,%1
      mov edx,%2
      int 80h
      mov eax,3
      mov ebx,2
      mov ecx,trash
      mov edx,1
      int 80h
%endmacro

section .text
      global _start

_start:
      write mss1,plen
      read n,1
      sub byte[n],'0'

      mov eax,'0'
      mov [j],eax
      call input
      call selection_sort
      write mss3,p3len
      call display

      mov eax,1
      mov ebx,0
      int 80h

input:
      write mss2,p2len
      mov [i],dword 0
loop1:
      mov esi,[i]
      cmp esi,[n]
      jge after1

      add esi,arr
      read esi,1

      inc dword[i]
      jmp loop1
after1:
      ret


display:
      write mss4,p4len
      write j,9
      write mss5,p5len
      mov [i],dword 0
loop2:
      mov esi,[i]
      cmp esi,[n]
      jge after2

      add esi,arr
      write esi,1
      write space,1

      inc dword[i]
      jmp loop2
after2:
      write newline,1
      ret


selection_sort:
      mov eax,0       ;al is
counter for outer loop,
initialise to 0
      mov bl,[n]
      sub bl,1        ;bl is n-
1
```

```
loop3:
      cmp al,bl         ;repeat
until al<n-1
      jge after3

      pushad
      call display
      popad

      mov ecx,0         ;clear
ecx register
      mov cl,al         ;ecx is
counter for inner loop,
initialise to al+1
      add cl,1

      mov edi,arr
      add edi,eax             ;edi
points to arr[eax]
loop4:
      cmp cl,[n]        ;repeat
until cl<n
      jge after4

      mov esi,arr
      add esi,ecx             ;esi
points to arr[ecx]
      mov bh,[esi]
      mov dh,[edi]
      cmp bh,dh
      jge after5        ;if bh<dh

      mov edi,arr             ;edi
points to arr[ecx]
      add edi,ecx
after5:
      inc cl
      jmp loop4
after4:
      ;; swap arr[eax] and
[edi]
      mov bh,[arr+eax]
      mov dh,[edi]
      mov [arr+eax],dh
      mov [edi],bh

      inc al
      inc byte[j]
      jmp loop3
after3:
      ret
```

OUTPUT

```
lloyd@LLOYD: /mnt/c/Users/lloyd/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:
5
6
1
3
2
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$
```