# 1]Write an ALP to implement bubble sort

# INPUT –

%macro print 2	minus db "-", 0
push eax	
push ebx	section .bss
push ecx	array resd 100
mov eax, 4	size resd 1
mov ebx, 1	buffer resb 12
mov ecx, %1	negative resb 1
mov edx, %2	swapped resb 1
int 0x80	
рор есх	section .text
pop ebx	global _start
pop eax	_start:
%endmacro	print prompt_size, 25
	call read_int
%macro exit 0	mov [size], eax
mov eax, 1	xor ebx, ebx
xor ebx, ebx	
int 0x80	input_loop:
%endmacro	cmp ebx, [size]
	jge input_done
section .data	print prompt_element, 15
prompt_size db "Enter number of elements: ", 0	call read_int
prompt_element db "Enter element: ", 0	mov [array + ebx*4], eax
msg_iteration db "Iteration ", 0	inc ebx
msg_array db "Array: ", 0	jmp input_loop
msg_complete db "Bubble sort executed", 10, 0	
msg_sorted db "Sorted array: ", 0	input_done:
newline db 10, 0	print newline, 1
space db " ", 0	mov ecx, [size]

```
dec ecx
                                                              cmp byte [swapped], 0
  jle sort_done
                                                              jne bubble_sort
bubble_sort:
                                                            sort_done:
  mov byte [swapped], 0
                                                              print newline, 1
  xor ebx, ebx
                                                              print msg_complete, 19
                                                              print newline, 1
inner_loop:
                                                              print msg_sorted, 14
  mov eax, [array + ebx*4]
                                                              call print_array
  cmp eax, [array + ebx*4 + 4]
                                                              print newline, 1
  jle no_swap
                                                              exit
  xchg eax, [array + ebx*4 + 4]
  mov [array + ebx*4], eax
                                                            read_int:
  mov byte [swapped], 1
                                                              push ebx
                                                              push ecx
                                                              push edx
no_swap:
                                                              push esi
  inc ebx
  cmp ebx, ecx
                                                              mov eax, 3
                                                              mov ebx, 0
  jl inner_loop
                                                              mov ecx, buffer
  push ecx
                                                              mov edx, 12
                                                              int 0x80
  print newline, 1
  print msg_iteration, 10
                                                              mov byte [negative], 0
  mov eax, [size]
                                                              mov esi, buffer
  sub eax, ecx
                                                              xor eax, eax
  call print_int
                                                              xor ebx, ebx
  print newline, 1
                                                              mov bl, [esi]
  print msg_array, 7
                                                              cmp bl, '-'
  call print_array
                                                              jne convert_start
  print newline, 1
                                                              mov byte [negative], 1
                                                              inc esi
  рор есх
```

convert\_start:

dec ecx

movzx edx, byte [esi] jns non\_negative inc esi push eax cmp dl, 10 print minus, 1 je convert\_done pop eax cmp dl, '0' neg eax jb convert\_start cmp dl, '9' non\_negative: ja convert\_start mov ecx, 10 sub dl, '0' mov edi, buffer imul eax, 10 add edi, 11 add eax, edx mov byte [edi], 0 jmp convert\_start mov esi, edi test eax, eax convert\_done: jnz digit\_loop cmp byte [negative], 1 dec edi jne positive\_num mov byte [edi], '0' jmp print\_num neg eax digit\_loop: positive\_num: dec edi pop esi pop edx xor edx, edx div ecx pop ecx add dl, '0' pop ebx mov [edi], dl ret test eax, eax jnz digit\_loop print\_int:

push ebx print\_num:

push ecx mov ecx, edi

push edx mov edx, esi

push esi sub edx, edi

push edi mov eax, 4

test eax, eax mov ebx, 1

push eax

int 0x80

print\_loop: print\_int\_done: cmp ebx, [size] jge print\_done pop edi mov eax, [array + ebx\*4] pop esi pop edx call print\_int рор есх print space, 1 pop ebx inc ebx pop eax jmp print\_loop ret print\_done: print\_array: рор есх push eax pop ebx push ebx pop eax push ecx ret xor ebx, ebx

#### OUTPUT -

```
root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp10# ./1
Enter number of elements:5
Enter element: 3
Enter element: 7
Enter element: 6
Enter element: 1
Enter element: 9
Iteration 1
Array: 3 6 1 7 9
Iteration 2
Array: 3 1 6 7 9
Iteration 3
Array: 1 3 6 7 9
Iteration 4
Array: 1 3 6 7 9
Bubble sort execute
Sorted array: 1 3 6 7 9
```

## 2]Write an ALP to implement insertion sort

### INPUT -%macro print 2 minus db "-", 0 push eax push ebx section .bss push ecx array resd 100 mov eax, 4 size resd 1 mov ebx, 1 buffer resb 12 mov ecx, %1 negative resb 1 mov edx, %2 key resd 1 int 0x80 рор есх section .text pop ebx global \_start pop eax \_start: %endmacro print prompt\_size, 25 call read\_int %macro exit 0 mov [size], eax mov eax, 1 xor ebx, ebx xor ebx, ebx int 0x80 input\_loop: %endmacro cmp ebx, [size] jge input\_done section .data print prompt\_element, 15 prompt\_size db "Enter number of elements: ", 0 call read\_int prompt\_element db "Enter element: ", 0 mov [array + ebx\*4], eax msg\_iteration db "Iteration ", 0 inc ebx msg\_array db "Array: ", 0 jmp input\_loop msg complete db "Insertion sort executed", 10, 0 input\_done: msg\_sorted db "Sorted array: ", 0 print newline, 1

mov ecx, 1

newline db 10, 0

space db " ", 0

```
mov [array + ebx*4 + 4], eax
```

jne convert\_start

inc esi

mov byte [negative], 1

insertion\_sort: cmp ecx, [size] inc ecx ; i++ jge sort\_done jmp insertion\_sort mov eax, [array + ecx\*4] sort\_done: mov [key], eax print newline, 1 mov ebx, ecx ; j = i print msg\_complete, 23 dec ebx ; j = i-1 print newline, 1 print msg\_sorted, 14 print newline, 1 call print\_array print msg\_iteration, 10 print newline, 1 exit mov eax, ecx call print\_int read\_int: print newline, 1 print msg\_array, 7 push ebx call print\_array push ecx print newline, 1 push edx push esi while\_loop: mov eax, 3 cmp ebx, 0 ; while j >= 0 mov ebx, 0 jl end\_while mov ecx, buffer mov eax, [array + ebx\*4] mov edx, 12 ; and array[j] > key cmp eax, [key] int 0x80 jle end\_while mov byte [negative], 0 mov esi, buffer mov edx, [array + ebx\*4]; array[j+1] = array[j]xor eax, eax mov [array + ebx\*4 + 4], edxxor ebx, ebx dec ebx ; j = j-1 mov bl, [esi] jmp while\_loop cmp bl, '-'

end\_while:

mov eax, [key]

push edi

convert\_start: test eax, eax

movzx edx, byte [esi] jns non\_negative

inc esi push eax

cmp dl, 10 print minus, 1

je convert\_done pop eax cmp dl, '0' neg eax

jb convert\_start

cmp dl, '9' non\_negative:

ja convert\_start mov ecx, 10

sub dl, '0' mov edi, buffer

imul eax, 10 add edi, 11

add eax, edx mov byte [edi], 0

jmp convert\_start mov esi, edi

test eax, eax

convert\_done: jnz digit\_loop

cmp byte [negative], 1 dec edi

jne positive\_num mov byte [edi], '0'

neg eax jmp print\_num

positive\_num: digit\_loop:

pop esi dec edi

pop edx xor edx, edx

pop ecx div ecx

pop ebx add dl, '0'

ret mov [edi], dl

test eax, eax

print\_int: jnz digit\_loop

push eax

push ebx print\_num:

push ecx mov ecx, edi

push edx mov edx, esi

push esi sub edx, ecx

```
mov eax, 4
                                                            xor ebx, ebx
  mov ebx, 1
  int 0x80
                                                          print_loop:
                                                            cmp ebx, [size]
print_int_done:
                                                            jge print_done
                                                            mov eax, [array + ebx*4]
  pop edi
                                                            call print_int
  pop esi
                                                            print space, 1
  pop edx
  рор есх
                                                            inc ebx
                                                            jmp print_loop
  pop ebx
  pop eax
  ret
                                                          print_done:
                                                            pop ecx
print_array:
                                                            pop ebx
  push eax
                                                            pop eax
  push ebx
                                                            ret
  push ecx
```

## OUTPUT -

```
{\tt root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp10\#./2}
Enter number of elements:6
Enter element: 12
Enter element: -5
Enter element: 3
Enter element: -12
Enter element: 1
Enter element: 4
Iteration 1
Array: 12 -5 3 -12 1 4
Iteration 2
Array: -5 12 3 -12 1 4
Iteration 3
Array: -5 3 12 -12 1 4
Iteration 4
Array: -12 -5 3 12 1 4
Iteration 5
Array: -12 -5 1 3 12 4
Insertion sort executed
Sorted array: -12 -5 1 3 4 12
```

## 3]Write an ALP to implement selection sort

## INPUT -%macro print 2 minus db "-", 0 push eax push ebx section .bss push ecx array resd 100 mov eax, 4 size resd 1 mov ebx, 1 buffer resb 12 mov ecx, %1 negative resb 1 mov edx, %2 min idx resd 1 int 0x80 рор есх section .text pop ebx global \_start pop eax \_start: %endmacro print prompt\_size, 25 call read\_int %macro exit 0 mov [size], eax mov eax, 1 xor ebx, ebx xor ebx, ebx int 0x80 input\_loop: %endmacro cmp ebx, [size] jge input\_done section .data print prompt\_element, 15 prompt\_size db "Enter number of elements: ", 0 call read\_int prompt\_element db "Enter element: ", 0 mov [array + ebx\*4], eax msg\_iteration db "Iteration ", 0 inc ebx msg\_array db "Array: ", 0 jmp input\_loop msg\_complete db "Selection sort executed", 10,

input\_done:

mov ecx, 0

print newline, 1

0

msg\_sorted db "Sorted array: ", 0

newline db 10, 0

space db " ", 0

jmp inner\_loop selection\_sort: cmp ecx, [size] swap\_elements: jge sort\_done mov eax, [array + ecx\*4] mov edx, [min\_idx] mov [min\_idx], ecx mov ebx, [array + edx\*4] mov ebx, ecx mov [array + ecx\*4], ebx inc ebx mov [array + edx\*4], eax print msg\_iteration, 10 inc ecx mov eax, ecx jmp selection\_sort inc eax sort\_done: call print\_int print msg\_complete, 23 print newline, 1 print msg\_array, 7 print newline, 1 print newline, 1 print msg\_sorted, 14 call print\_array call print\_array print newline, 1 print newline, 1 print newline, 1 exit

inner_loop:	read_int:
cmp ebx, [size]	push ebx
jge swap_elements	push ecx
	push edx
mov eax, [array + ebx*4]	push esi
mov edx, [min_idx]	mov eax, 3
mov edx, [array + edx*4]	mov ebx, 0
cmp eax, edx	mov ecx, buffer
jge skip_update	mov edx, 12
mov [min_idx], ebx	int 0x80

skip\_update:

inc ebx

mov byte [negative], 0

mov esi, buffer

xor eax, eax

xor ebx, ebx print\_int: mov bl, [esi] push eax cmp bl, '-' push ebx jne convert\_start push ecx mov byte [negative], 1 push edx inc esi push esi push edi convert\_start: test eax, eax movzx edx, byte [esi] jns non\_negative inc esi push eax cmp dl, 10 print minus, 1 je convert\_done pop eax cmp dl, '0' neg eax jb convert\_start cmp dl, '9' non\_negative: ja convert\_start mov ecx, 10 sub dl, '0' mov edi, buffer imul eax, 10 add edi, 11 mov byte [edi], 0 add eax, edx jmp convert\_start mov esi, edi test eax, eax convert\_done: jnz digit\_loop cmp byte [negative], 1 dec edi mov byte [edi], '0' jne positive\_num neg eax jmp print\_num positive\_num: digit\_loop: pop esi dec edi pop edx xor edx, edx pop ecx div ecx add dl, '0' pop ebx

ret

mov [edi], dl

test eax, eax

jnz digit\_loop push eax push ebx print\_num: push ecx mov ecx, edi xor ebx, ebx mov edx, esi sub edx, ecx print\_loop: mov eax, 4 cmp ebx, [size] jge print\_done mov ebx, 1 int 0x80 mov eax, [array + ebx\*4] call print\_int print\_int\_done: print space, 1 pop edi inc ebx pop esi jmp print\_loop pop edx print\_done: рор есх pop ebx pop ecx pop eax pop ebx ret pop eax ret

print\_array:

#### OUTPUT -

```
root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp10# ./3
Enter number of elements:7
Enter element: 9
Enter element: 8
Enter element: 7
Enter element: 6
Enter element: 5
Enter element: 4
Enter element: 3
Iteration 1
Array:
9876543
Iteration 2
Array:
3 8 7 6 5 4 9
Iteration 3
Array:
3 4 7 6 5 8 9
Iteration 4
Array:
3 4 5 6 7 8 9
Iteration 5
Array:
3 4 5 6 7 8 9
Iteration 6
Array:
3 4 5 6 7 8 9
Iteration 7
Array:
3 4 5 6 7 8 9
Selection sort executed
Sorted array: 3 4 5 6 7 8 9
```

**CONCLUSION – Sorting algorithms were successfully implemented using NASM.**