COAL PROJECT REPORT

Sortify Library

TEAM MEMBERS:

Aarib Ahmed 22K-4004

Partham Kumar 22K-4079

Arham Hussain 22k-4080

Introduction:

The SL (Sortify Library) project aims to provide a comprehensive solution for sorting data efficiently and effectively.

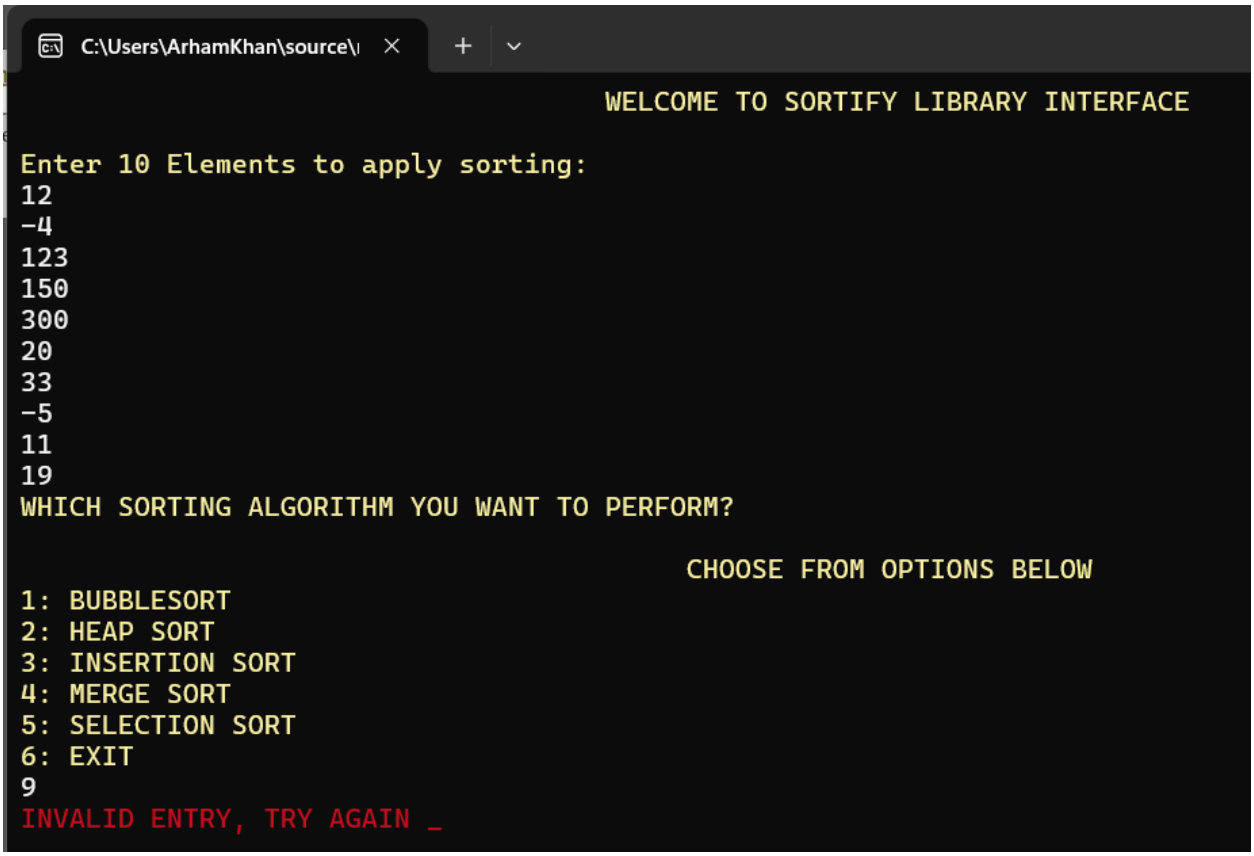
Resources:

Website for using macros.inc library

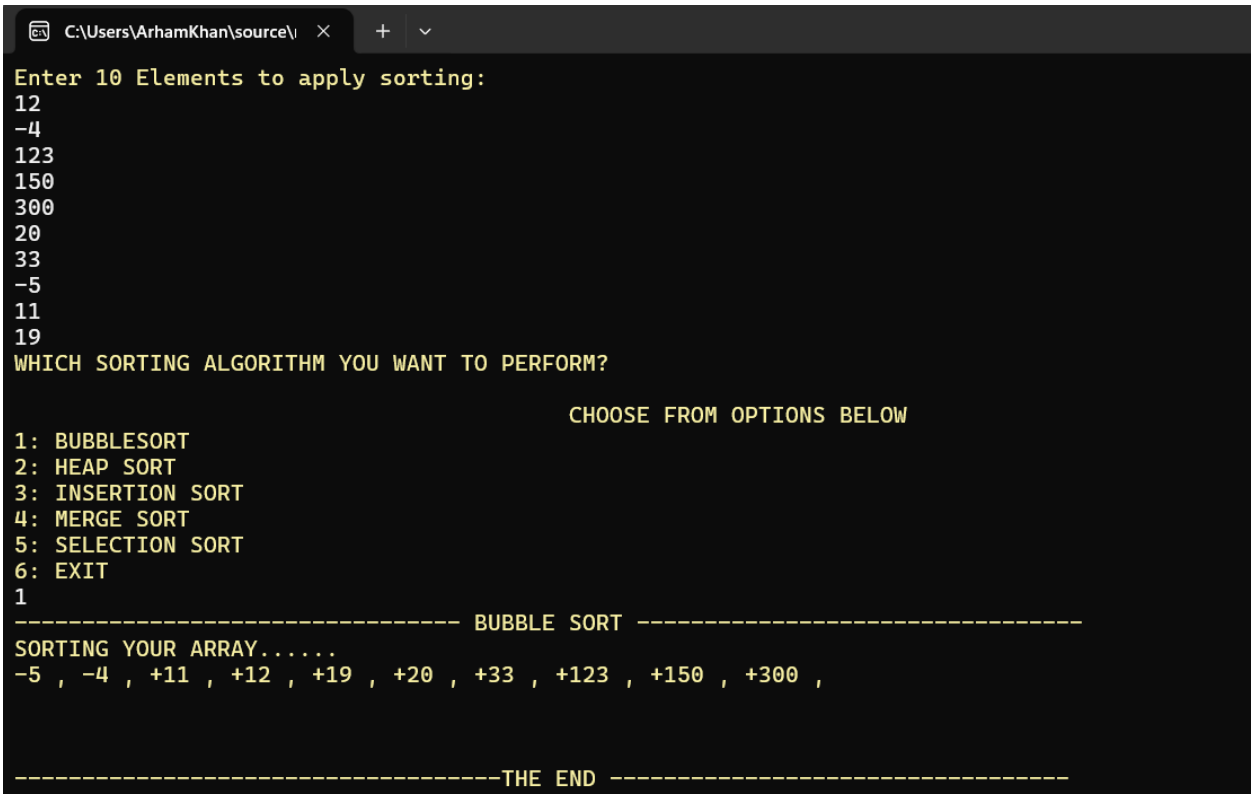
https://users.cs.fiu.edu/~downeyt/cop3402/irvine/MACROS.INC

Code Outputs:

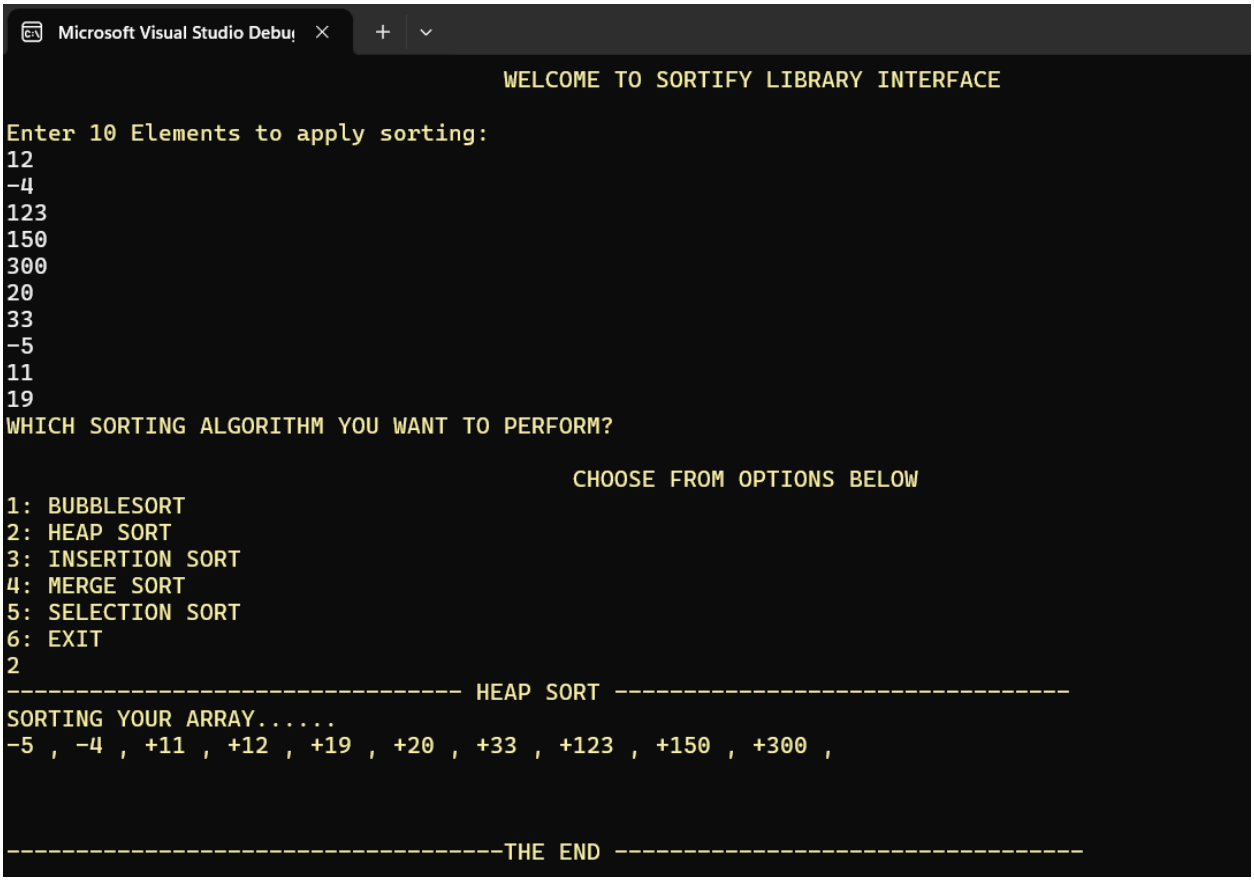
Output for invalid entry:



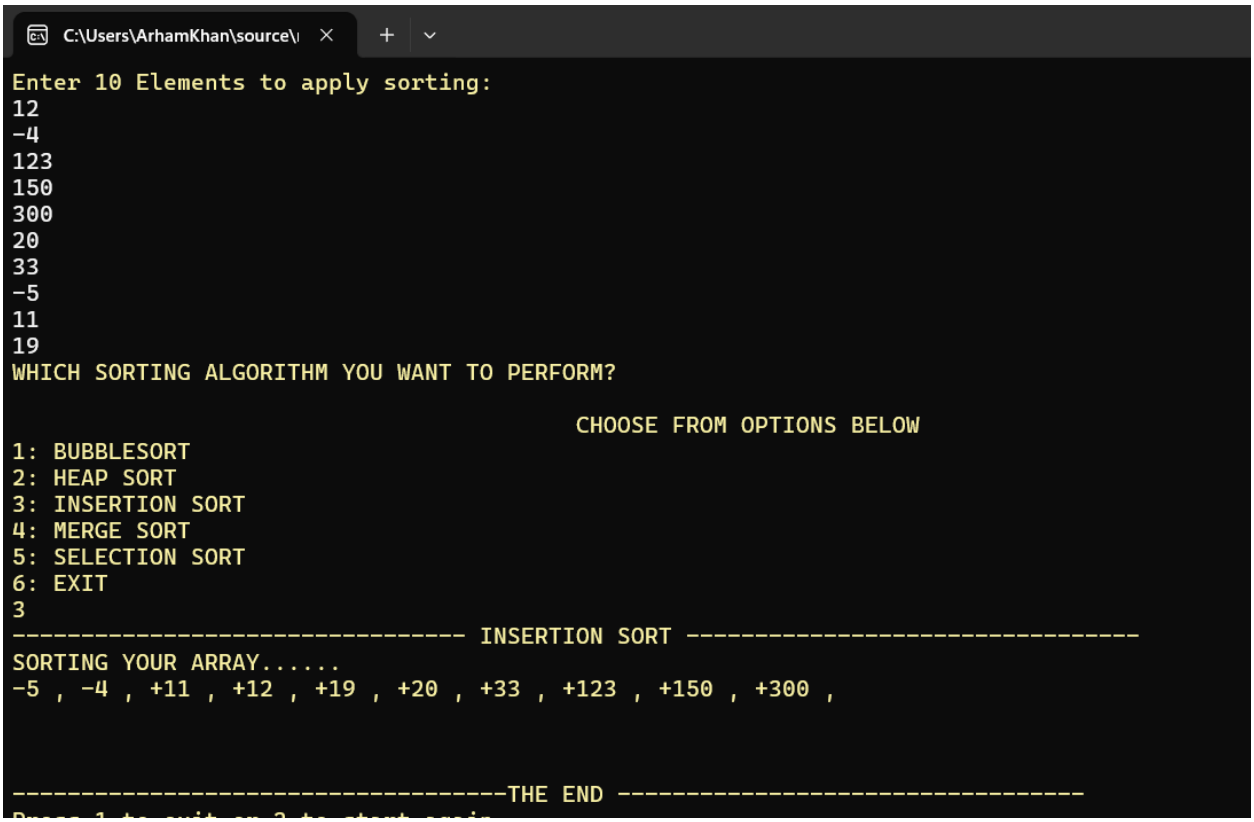
1. **Bubble Sort output:**



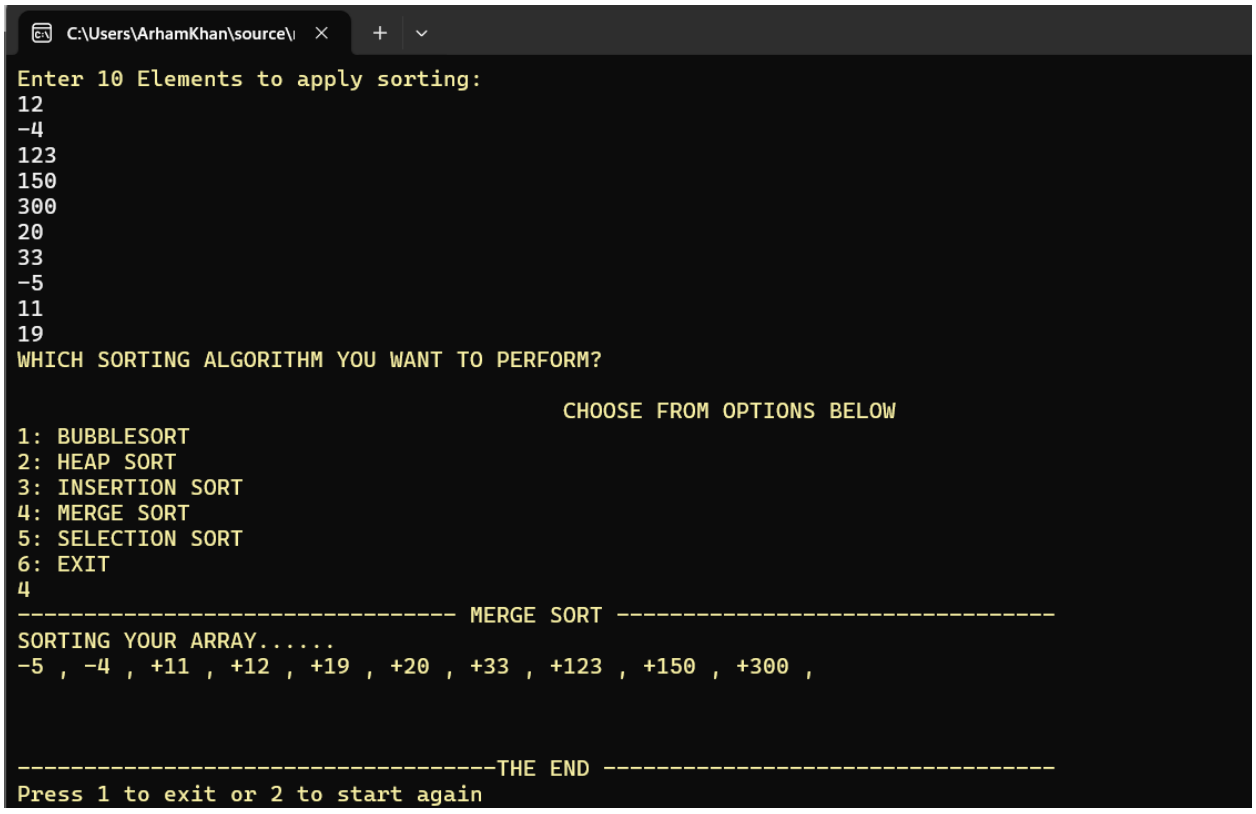
2. Heap Sort output:



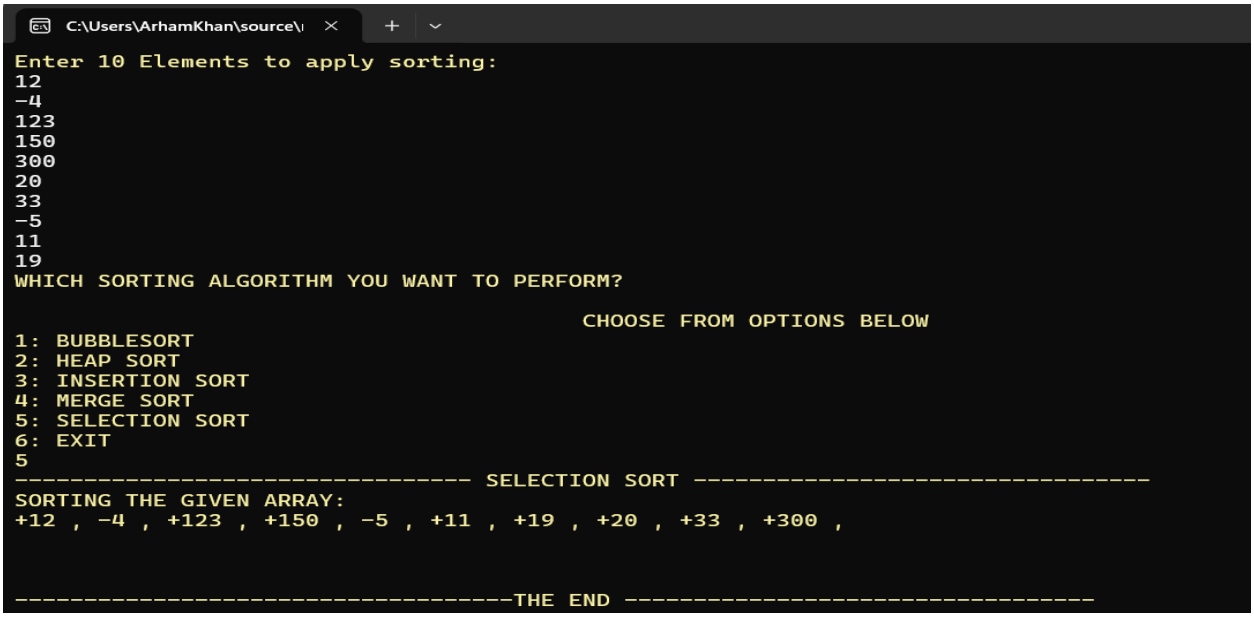
3. Insertion Sort output:



4. Merge Sort Output:



5. Selection Sort Output:



CODE:

```
INCLUDE Irvine32.inc
INCLUDE macros.inc
;Macros.inc are custom macros defined by a programmer (using it just for displaying
prompts)
.data
arr dword 10 dup(?)
comma byte " , ", 0
prompt byte "Press 1 to exit or 2 to start again ",0

.code
main PROC
mov eax,0

start_program:
call clrscr
mov eax, yellow + (black*16)
call settextcolor
mwrite"                                    WELCOME TO SORTIFY LIBRARY INTERFACE"
mov eax, white + (black*16)
call settextcolor
call crlf
call crlf


mov eax, yellow + (black*16)
call settextcolor
mwrite"Enter 10 Elements to apply sorting: "
mov eax, white + (black*16)
call settextcolor
call crlf
mov edx, offset arr
mov ecx,lengthof arr
arr_loop:
call readint
mov [edx], eax
add edx, 4
loop arr_loop


mov eax, yellow + (black*16)
call settextcolor
mwrite"WHICH SORTING ALGORITHM YOU WANT TO PERFORM?"
mov eax, white + (black*16)
call settextcolor
call crlf
call crlf
mov eax, yellow + (black*16)
call settextcolor
MWRITE"                                          CHOOSE FROM OPTIONS BELOW"
call crlf
MWRITE"1: BUBBLESORT"
call crlf
MWRITE"2: HEAP SORT"
call crlf
```

```
MWRITE"3: INSERTION SORT"
call crlf
MWRITE"4: MERGE SORT"
call crlf
MWRITE"5: SELECTION SORT"
call crlf
MWRITE"6: EXIT"

menu_loop:
call crlf
call readint

cmp eax,1
je bubble
cmp eax,2
je heap
cmp eax,3
je insertion
cmp eax,4
je merge
cmp eax,5
je selection
cmp eax, 6
je exit_program
mov eax, red + (black*16)
call settextcolor
mwrite"INVALID ENTRY, TRY AGAIN _"
mov eax, white + (black*16)
call settextcolor
jg menu_loop

bubble:
mov eax, yellow + (black*16)
call settextcolor
mwrite"-------------------------------- BUBBLE SORT ------------------------------
--"
call crlf
mwrite"SORTING YOUR ARRAY......"
mov esi,offset arr
mov ebx,lengthof arr
call BubbleSort
mov esi, offset arr
mov edx, offset comma
mov ecx, lengthof arr
call crlf
l1000:
mov eax, [esi]
call writeint
call writestring
add esi,4
loop l1000

jmp l100

heap:
mwrite"-------------------------------- HEAP SORT --------------------------------
"
call crlf
```

```
mwrite"SORTING YOUR ARRAY......"
mov eax,type arr
mov esi,offset arr
mov edx,lengthof arr
call HeapSort
mov esi, offset arr
mov edx, offset comma
mov ecx, lengthof arr
call crlf
l2000:
mov eax, [esi]
call writeint
call writestring
add esi,4
loop l2000
jmp l100

insertion:
mwrite"--------------------------------- INSERTION SORT ----------------------------
-----"
call crlf
mwrite"SORTING YOUR ARRAY......"
mov ebx,type arr
mov esi,offset arr
mov ecx,lengthof arr
call Insertion_sort
mov esi, offset arr
mov edx, offset comma
mov ecx, lengthof arr
call crlf
l3000:
mov eax, [esi]
call writeint
call writestring
add esi,4
loop l3000
jmp l100

merge:
mwrite"--------------------------------- MERGE SORT ---------------------------------
-"
call crlf
mwrite"SORTING YOUR ARRAY......"
mov eax,type arr
mov esi,offset arr
mov ebx,0
mov edx,lengthof arr
dec edx
call MergeSort
mov esi, offset arr
mov ecx, lengthof arr
mov edx, offset comma
call crlf
l4000:
mov eax, [esi]
call writeint
call writestring
add esi,4
```

```
loop l4000

jmp l100

selection:
mwrite"--------------------------------- SELECTION SORT ----------------------------
-----"
call crlf
mwrite"SORTING THE GIVEN ARRAY:"
mov eax,type arr
mov esi,offset arr
mov ecx,lengthof arr
call SelectionSort
mov esi, offset arr
mov edx, offset comma
mov ecx, lengthof arr
call crlf

l5000:
mov eax, [esi]
call writeint
call writestring
add esi,4
loop l5000
jmp l100

l100:
call crlf
call crlf
call crlf
call crlf
mwrite"-----------------------------------THE END ---------------------------------
-"
call crlf
mov edx, offset prompt
call writestring
call crlf
l2:
mov eax, 0
call readint
cmp eax, 1
je exit_program
cmp eax, 2
je start_program
mov eax, red + (black*16)
call settextcolor
mwrite"INVALID ENTRY, TRY AGAIN"
mov eax, yellow + (black*16)
call settextcolor
jne l2


exit_program:
mov eax, white + (black*16)
call settextcolor
exit
main ENDP
```

```
BubbleSort proc
mov edi,esi
mov ecx,ebx
dec ecx
mov ebx,0
mov eax,0
outerLoop:
push ecx
mov esi,edi
innerLoop:
mov eax,[esi]
mov ebx,[esi + 4]
cmp eax,ebx
jg swapElements
continueLoop:
add esi,4
loop innerLoop
pop ecx
loop outerLoop
jmp endProgram
swapElements:
mov eax,[esi]
mov ebx,[esi + 4]
xchg eax,ebx
mov [esi],eax
mov [esi + 4],ebx
jmp continueLoop
endProgram:
ret
BubbleSort ENDP


;------------------------------------ HEAP-SORT FUNTION -------------------------
-------------
HeapSort PROC
push eax
push edx
mov eax,edx
mov edx,0
mov ebx,2
div ebx
cmp edx,0
jnz isOdd
jmp isEven
isOdd:
pop edx
dec edx
jmp nextComparison
isEven:
pop edx
nextComparison:
pop eax
cmp eax,4
jz ISDWORD
ISDWORD:
call DHeapSort
```

```
ret
HeapSort ENDP

DHeapSort PROC
local i:DWORD,n:DWORD,temp:DWORD
mov n,edx
shr edx,1
dec edx
mov i,edx
mov ecx,0
buildHeap:
mov ebx,i
mov edx,n
call DHeapify
mov eax,0
dec eax
dec i
cmp eax,i
jz goOut
loop buildHeap
goOut:
mov eax,n
mov i,eax
dec i
mov ecx,i
ExtractAndSort:
mov eax,[esi]
mov ebx,i
xchg eax,[esi+ebx*4]
mov [esi],eax
mov ebx,0
mov edx,i
mov temp,ecx
call DHeapify
mov ecx,temp
dec i
loop ExtractAndSort
ret
DHeapSort ENDP
; ESI = OFFSET of Array
; EDX = END
; EBX = START
DHeapify PROC
LOCAL largest:DWORD,left:DWORD,right:DWORD,n:DWORD,i:DWORD
mov i,ebx
mov n,edx
mov largest,ebx
shl ebx,1
mov left,ebx
mov right,ebx
inc left
add right,2
mov eax,left
cmp eax,n
jl checkSecondConditionForLargestLeft
jmp checkSecondIfCondition
checkSecondConditionForLargestLeft:
mov ebx,left
```

```
mov ecx,largest
mov eax,[esi + ebx*4]
cmp eax,[esi + ecx*4]
jg largestLeftExists
jmp checkSecondIfCondition
largestLeftExists:
mov largest,ebx
checkSecondIfCondition:
mov eax,right
cmp eax,n
jl checkSecondConditionForLargestRight
jmp checkThirdCondition
checkSecondConditionForLargestRight:
mov ebx,right
mov ecx,largest
mov eax,[esi + ebx * 4]
cmp eax,[esi + ecx * 4]
jg largestRightExists
jmp checkThirdCondition
largestRightExists:
mov largest,ebx
checkThirdCondition:
mov eax,i
cmp eax,largest
jnz RecurseHeapify
jmp returnFromFunction
RecurseHeapify:
mov eax,i
mov ebx,largest
mov ecx,[esi + eax*4]
xchg ecx,[esi + ebx*4]
mov [esi + eax*4],ecx
mov edx,n
mov ebx,largest
call DHeapify
returnFromFunction:
ret
DHeapify ENDP

;------------------------------------ INSERTION FUNTION -------------------------
----------
Insertion_sort PROC USES eax ecx edx
LOCAL x:DWORD, i:DWORD, j:DWORD,lengthArr:DWORD
mov eax, 1
mov i, eax
mov eax, 0
mov j, eax
mov lengthArr,ecx
mov ecx, lengthArr
dec ecx
D1:
push ecx
mov ecx, ebx
mov edx, i
mov eax, [esi+edx*4]
mov x, eax
push ecx
mov ecx, 0
```

```
mov eax, i
mov j, eax
dec j
mov ebx, 0
D2:
mov ebx, 0
mov edx, j
cmp j, ebx
JL D3
mov ebx, [esi+edx*TYPE esi]
cmp ebx, x
JLE D3
inc edx
mov [esi+edx*TYPE esi], ebx
dec j
loop D2
D3:
pop ecx
mov eax, x
mov edx, j
inc edx
mov [esi+edx*TYPE esi], eax
inc i
pop ecx
loop D1
return:
ret
Insertion_sort ENDP

;------------------------------------ MERGE-SORT FUNTION ----------------------
MergeSort PROC
LOCAL mid:DWORD,lower:DWORD,higher:DWORD,typeArr:DWORD
mov typeArr,eax
mov lower,ebx
mov higher,edx
cmp ebx,edx
jl recurse
jmp endProgram
recurse:
mov eax,lower
add eax,higher
shr eax,1
mov mid,eax
mov ebx,lower
mov edx,mid
mov eax,typeArr
call MergeSort
mov ebx,mid
inc ebx
mov edx,higher
mov eax,typeArr
call MergeSort
mov ebx,lower
mov edx,higher
mov eax,typeArr
cmp eax,4
jz isDWORD
isDWORD:
```

```
mov eax,mid
call DMerge
jmp endProgram

endProgram:
ret
MergeSort ENDP

DMerge PROC
LOCAL i:DWORD,j:DWORD,k:DWORD,lower:DWORD,mid:DWORD,higher:DWORD,a[100]:SDWORD
mov i,ebx
mov k,ebx
mov j,eax
inc j
mov mid,eax
mov higher,edx
mov lower,ebx
mov ecx,0
startLoop:
mov edx,mid
cmp i,edx
jg outsideFirst
mov edx,higher
cmp j,edx
jg outsideFirst
mov ebx,i
mov eax,[esi + ebx * 4]
mov ebx,j
cmp eax,[esi + ebx * 4]
jl firstConditionTrue
jmp secondConditionTrue
firstConditionTrue:
mov ebx,i
mov edx,k
mov eax,[esi + ebx * 4]
mov [a + edx * 4],eax
inc k
inc i
jmp continueLoop
secondConditionTrue:
mov ebx,j
mov edx,k
mov eax,[esi + ebx * 4]
mov [a + edx * 4],eax
inc k
inc j
continueLoop:
loop startLoop
outsideFirst:
mov ecx,0
secondLoop:
mov eax,mid
cmp i,eax
jg outsideSecond
mov ebx,i
mov edx,k
mov eax,[esi + ebx * 4]
mov [a + edx * 4],eax
```

```
inc k
inc i
loop secondLoop
outsideSecond:
thirdLoop:
mov eax,higher
cmp j,eax
jg outsideThird
mov ebx,j
mov edx,k
mov eax,[esi + ebx * 4]
mov [a + edx * 4],eax
inc k
inc j
loop thirdLoop
outsideThird:
mov eax,lower
mov i,eax
mov ecx,0
finalLoop:
mov eax,k
cmp i,eax
jge outsideFinal
mov ebx,i
mov eax,[a + ebx * 4]
mov [esi + ebx * 4],eax
inc i
loop finalLoop
outsideFinal:
ret
DMerge ENDP

;------------------------------------ SELECTION-SORT FUNTION ---------------------
--
SelectionSort PROC
LOCAL i:DWORD,j:DWORD,min:DWORD,sizeOfArray:DWORD
mov eax,4
mov sizeOfArray,ecx
mov i,eax
mov j,eax
mov min,eax
mov ecx,sizeOfArray
dec ecx
outerLoop:
mov ebx,i
mov min,ebx
push ecx
mov edx,i
mov j,edx
innerLoop:
inc j
mov edx,j
mov eax,[esi + edx * 4]
mov edx,min
mov ebx,[esi + edx * 4]
cmp eax,ebx
jl markNewMin
jmp continueLoop
```

```
markNewMin:
mov edx,j
mov min,edx
continueLoop:
loop innerLoop
mov eax,i
mov ebx,[esi + eax * 4]
mov edx,min
xchg ebx,[esi + edx * 4]
mov [esi + eax * 4],ebx
pop ecx
inc i
loop outerLoop
ret
SelectionSort ENDP
END MAIN
```