**Coal Assignment 5**

*Ahmed Kasteer*

*20F-0336*

*Section 3D*

Question 1

INCLUDE Irvine32.inc

.data

arr BYTE "H","E","L","L","O"

Rowsize = ($-arr)

BYTE "B","I","M","C","D"

BYTE " "," "," "," ","E"

BYTE "G","J","K","L","M"

BYTE "N","O","P","Q","R"

stringname BYTE 5 dup (?)

find BYTE ?

row DWORD ?

column DWORD ?

row1 DWORD ?

column1 DWORD ?

counter DWORD 0

rowIndex DWORD 0

columnIndex DWORD 0

pointer DWORD ?

tempvar SDWORD ?

D1 BYTE "Enter string to encrypt: ",0

D2 BYTE "Repeated elemnts: ",0

D3 BYTE "Ecrypted string: ",0

.code

main PROC

call input\_string

mov ecx,2

mov esi,0

l1:

mov pointer,esi

mov al,stringname[esi]

mov find,al

call search

mov eax,rowIndex

mov row,eax

mov eax,columnIndex

mov column,eax

mov al,stringname[esi+1]

mov find,al

call search

mov eax,rowIndex

mov row1,eax

mov eax,columnIndex

mov column1,eax

call case\_checker

add esi,2

loop l1

call crlf

mov edx,offset D3

call writestring

mov edx,offset stringname

call writestring

exit

main ENDP

input\_string proc uses ecx esi

F1:

mov edx,offset D1

call writestring

mov edx,offset stringname

mov ecx,sizeof stringname

call ReadString

mov ecx,4

mov esi,0

F2:

mov al,stringname[esi]

cmp al,0

je F1

inc esi

loop F2

mov ecx,3

mov esi,0

F3:

mov al,stringname[esi]

mov edx,esi

inc edx

mov ebx,ecx

F4:

.if ( al == stringname[edx] )

mov edx,offset D2

call writestring

jmp F1

.endif

inc edx

loop F4

inc esi

mov ecx,ebx

loop F3

ret

input\_string endp

search proc uses ecx esi

mov rowIndex,0

mov columnIndex,0

mov ecx, 5

L1:

mov counter, ecx

mov ecx, RowSize

L2:

mov ebx, offset arr

mov eax, rowIndex

mov edx, RowSize

mul edx

add ebx, eax

mov esi, columnIndex

mov eax, 0

mov al, [ebx + esi \* TYPE arr]

.if (al == find)

jmp endlabel

.endif

inc columnIndex

loop L2

inc rowIndex

mov columnIndex, 0

mov ecx, counter

loop L1

endlabel:

ret

search endp

case\_checker proc

mov eax,row

mov ebx,column

.if eax == row1

call case1

.elseif ebx == column1

call case2

.else

call case3

.endif

ret

case\_checker endp

case1 proc uses ecx esi

mov ecx,column

mov ebx,column1

mov eax,RowSize

mov edx,row

mul edx

.if (ecx == 4)

mov ecx,0

.else

inc ecx

.endif

.if (ebx == 4)

mov ebx,0

.else

inc ebx

.endif

mov dl,arr[eax + ecx \* TYPE arr]

mov dh,arr[eax + ebx \* TYPE arr]

mov stringname[esi],dl

inc esi

mov stringname[esi],dh

ret

case1 endp

case2 proc uses ecx esi

mov ecx,column

mov eax,RowSize

mov edx,row

.if (edx == 4)

mov edx,0

.else

inc edx

.endif

mul edx

mov ebx,eax

mov eax,RowSize

mov edx,row1

.if (edx == 4)

mov edx,0

.else

inc edx

.endif

mul edx

mov dl,arr[ebx + ecx \* TYPE arr]

mov dh,arr[eax + ecx \* TYPE arr]

mov stringname[esi],dl

inc esi

mov stringname[esi],dh

ret

case2 endp

case3 proc uses ecx esi

mov edx,column

sub edx,column1

call abs

mov ecx,column

cmp ecx,column1

jg row\_greater

jl row1\_greater

row\_greater:

sub ecx,edx

add column1,edx

jmp endl

row1\_greater:

add ecx,edx

sub column1,edx

jmp endl

endl:

mov eax,RowSize

mov ebx,row1

mul ebx

mov ebx,eax

mov eax,RowSize

mov edi,row

mul edi

mov edi,column1

mov dl,arr[eax + ecx \* TYPE arr]

mov dh,arr[ebx + edi \* TYPE arr]

mov stringname[esi],dl

inc esi

mov stringname[esi],dh

ret

case3 endp

abs proc uses ecx

mov tempvar,edx

.if (tempvar < 0)

mov edx,tempvar

add edx,-1

mov ecx,11111111111111111111111111111111b

xor edx,ecx

.endif

ret

abs endp

end main

Text

Description automatically generated

Question 2

INCLUDE Irvine32.inc

.data

arr BYTE 0,0,0,0

RowSize = ($-arr)

BYTE 0,0,0,0

BYTE 0,0,0,0

BYTE 0,0,0,0

rowindex DWORD 0

columnIndex DWORD 0

D1 BYTE "Enter a 4 digit value: ",0

D2 BYTE "Reversed: ",0

.code

main PROC

mov edx,offset D1

call writestring

call readhex

mov edx,eax

call Input

mov edx,0

call Output

mov eax,edx

mov edx,offset D2

call writestring

call writehex

exit

main ENDP

Input proc

mov rowindex,0

mov columnIndex,0

mov ecx,RowSize

l1:

mov ebx,ecx

mov ecx,RowSize

l2:

mov eax,RowSize

push edx

mov edi,rowindex

mul edi

mov edi,eax

pop edx

mov esi,columnIndex

shr edx,1

jc Set

jnc NotSet

Set:

mov arr[edi + esi],1

jmp endlabel

NotSet:

mov arr[edi + esi],0

jmp endlabel

endlabel:

inc columnIndex

loop l2

mov columnIndex,0

mov ecx,ebx

inc rowindex

loop l1

ret

Input endp

Output proc

mov rowindex,0

mov columnIndex,0

mov ecx,RowSize

l1:

mov ebx,ecx

mov ecx, RowSize

l2:

mov eax,RowSize

push edx

mov edi,rowindex

mul edi

mov edi,eax

pop edx

mov esi,columnIndex

cmp arr[edi+esi],1

je isOne

jne isNotOne

isOne:

stc

rcl edx,1

jmp endlabel

isNotOne:

clc

rcl edx,1

jmp endlabel

endlabel:

inc columnIndex

loop l2

mov columnIndex,0

mov ecx,ebx

inc rowindex

loop l1

ret

Output endp

END main

Text

Description automatically generated