**COAL Assignment 03**

**Question #01**

INCLUDE Irvine32.inc

.data

dividend DWORD 0D4A4h

divisor DWORD 0Ah

result DWORD ?

remainder DWORD ?

.code

RecDivision PROC

; base case: if dividend <= 5h

mov eax, [ebp+8] ; load dividend from stack

cmp eax, 05h

jle EndRec

mov ebx, [ebp+12] ; load divisor from stack

div ebx

push eax ; push quotient as new dividend

push ebx ; push divisor again

call RecDivision

EndRec:

ret

RecDivision ENDP

main PROC

mov eax, dividend

mov ebx, divisor

push ebx

push eax

call RecDivision

exit

main ENDP

END main

A screenshot of a computer program

Description automatically generated

**Question #02**

INCLUDE Irvine32.inc

.data

intArray DWORD 12, 45, 67, 89, 23, 56, 34, 90, 11, 76, 42, 18, 63, 95, 7, 50, 33, 81, 29, 64

ARRAY\_SIZE = ($ - intArray) / TYPE intArray

enterMsg BYTE "Enter an integer to search for: ", 0

foundMsg BYTE "Value found at index: ", 0

notFoundMsg BYTE "Value not found in the array.", 0

.code

RecSearch PROC USES ebx ecx edx esi, ptrArray:PTR DWORD, target:DWORD, currIndex:DWORD, arrSize:DWORD

; Base case: index out of bounds

mov eax, currIndex

cmp eax, arrSize

jge notFoundd

; check current element

mov esi, ptrArray

mov ecx, currIndex

mov edx, [esi + ecx \* 4]

cmp edx, target

je foundd

inc eax

INVOKE RecSearch, ptrArray, target, eax, arrSize

ret

foundd:

mov eax, currIndex

ret

notFoundd:

mov eax, -1

ret

RecSearch ENDP

main PROC

mov edx, offset enterMsg

call WriteString

call ReadInt

mov ebx, eax

INVOKE RecSearch, ADDR intArray, ebx, 0, ARRAY\_SIZE

; check search result

cmp eax, -1

je ValueNotFound

push eax

mov edx, offset foundMsg

call WriteString

pop eax

call WriteInt

jmp endd

ValueNotFound:

mov edx, offset notFoundMsg

call WriteString

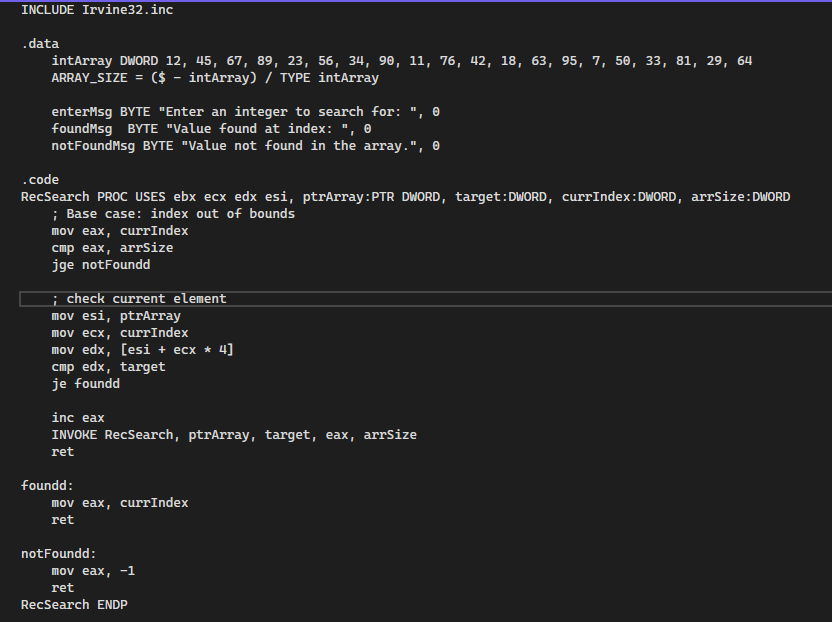
endd:

call crlf

exit

main ENDP

END main



A screenshot of a computer program

Description automatically generated

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

**Question #03**

INCLUDE Irvine32.inc

.data

sourceStr BYTE "This is the source string", 0

targetStr BYTE SIZEOF sourceStr DUP(?)

sourceMsg BYTE "Source String: ", 0

targetMsg BYTE "Target String: ", 0

.code

UniqueStringCopy PROC

push esi

push edi

push ebx

push ecx

; initialize pointers

mov esi, offset sourceStr

mov edi, offset targetStr

NextChar:

mov al, [esi] ; load the current character

cmp al, 0 ; check for null terminator

je EndCopy ; exit if end of source string

cmp al, ' ' ; skip spaces

je SkipChar

; check if character is unique

mov ebx, offset targetStr

; check the target string, if current char is not found here then it's unique.

CheckUnique:

cmp BYTE PTR [ebx], 0

je AddChar

cmp BYTE PTR [ebx], al

je SkipChar

inc ebx ; move to next character in target string

jmp CheckUnique

AddChar:

mov [edi], al ; add unique character to target string

inc edi ; move target pointer forward

SkipChar:

inc esi ; move to next character in source string

jmp NextChar

EndCopy:

mov BYTE PTR [edi], 0

pop ecx

pop ebx

pop edi

pop esi

ret

UniqueStringCopy ENDP

main PROC

mov edx, offset sourceMsg

call WriteString

mov edx, offset sourceStr

call WriteString

call crlf

call UniqueStringCopy

mov edx, offset targetMsg

call WriteString

mov edx, offset targetStr

call WriteString

call crlf

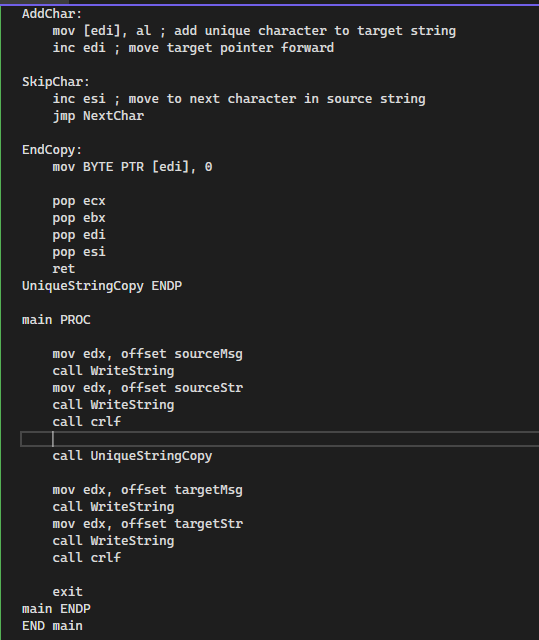
exit

main ENDP

END main

A screenshot of a computer program

Description automatically generated

A black background with white text

Description automatically generated

**Question #04**

INCLUDE Irvine32.inc

.data

MAX\_SIZE = 100

inputMsg BYTE "Enter a string: ", 0

inputBuffer BYTE MAX\_SIZE DUP(?)

resultMsg BYTE "Vowel counts:", 0

vowelLabels BYTE "aeiou", 0

vowelCounts BYTE 5 DUP(0)

aMsg BYTE "a/A: ", 0

eMsg BYTE "e/E: ", 0

iMsg BYTE "i/I: ", 0

oMsg BYTE "o/O: ", 0

uMsg BYTE "u/U: ", 0

.code

CountVowels PROC

mov esi, offset inputBuffer ; Load the input string address

mov ebx, offset vowelCounts ; Load the vowel counts array address

StartCount:

; load current character from input string

mov al, BYTE PTR [esi]

; check for end of string (null terminator)

cmp al, 0

je Done

; check for each vowel directly

cmp al, 'a'

je IncrementA

cmp al, 'A'

je IncrementA

cmp al, 'e'

je IncrementE

cmp al, 'E'

je IncrementE

cmp al, 'i'

je IncrementI

cmp al, 'I'

je IncrementI

cmp al, 'o'

je IncrementO

cmp al, 'O'

je IncrementO

cmp al, 'u'

je IncrementU

cmp al, 'U'

je IncrementU

jmp NextChar ; if not a vowel, continue with the next character

IncrementA:

inc BYTE PTR [ebx]

jmp NextChar

IncrementE:

inc BYTE PTR [ebx+1]

jmp NextChar

IncrementI:

inc BYTE PTR [ebx+2]

jmp NextChar

IncrementO:

inc BYTE PTR [ebx+3]

jmp NextChar

IncrementU:

inc BYTE PTR [ebx+4]

jmp NextChar

NextChar:

inc esi

jmp StartCount

Done:

ret

CountVowels ENDP

PrintVowelCounts PROC

mov ecx, 5

mov esi, offset vowelCounts

mov edi, offset vowelLabels

PrintLoop:

push ecx

mov al, BYTE PTR [edi]

; print corresponding message for the vowel

cmp al, 'a'

je PrintA

cmp al, 'e'

je PrintE

cmp al, 'i'

je PrintI

cmp al, 'o'

je PrintO

cmp al, 'u'

je PrintU

PrintA:

mov edx, offset aMsg

call WriteString

movzx eax, BYTE PTR [esi]

call WriteInt

call Crlf

jmp NextVowel

PrintE:

mov edx, offset eMsg

call WriteString

movzx eax, BYTE PTR [esi]

call WriteInt

call Crlf

jmp NextVowel

PrintI:

mov edx, offset iMsg

call WriteString

movzx eax, BYTE PTR [esi]

call WriteInt

call Crlf

jmp NextVowel

PrintO:

mov edx, offset oMsg

call WriteString

movzx eax, BYTE PTR [esi]

call WriteInt

call Crlf

jmp NextVowel

PrintU:

mov edx, offset uMsg

call WriteString

movzx eax, BYTE PTR [esi]

call WriteInt

call Crlf

jmp NextVowel

NextVowel:

inc esi

inc edi

pop ecx

dec ecx

jnz PrintLoop

ret

PrintVowelCounts ENDP

main PROC

; clear vowel counts

mov edi, offset vowelCounts

mov ecx, 5

xor eax, eax

rep stosb

mov edx, offset inputMsg

call WriteString

mov edx, offset inputBuffer

mov ecx, MAX\_SIZE

call ReadString

call CountVowels

mov edx, offset resultMsg

call WriteString

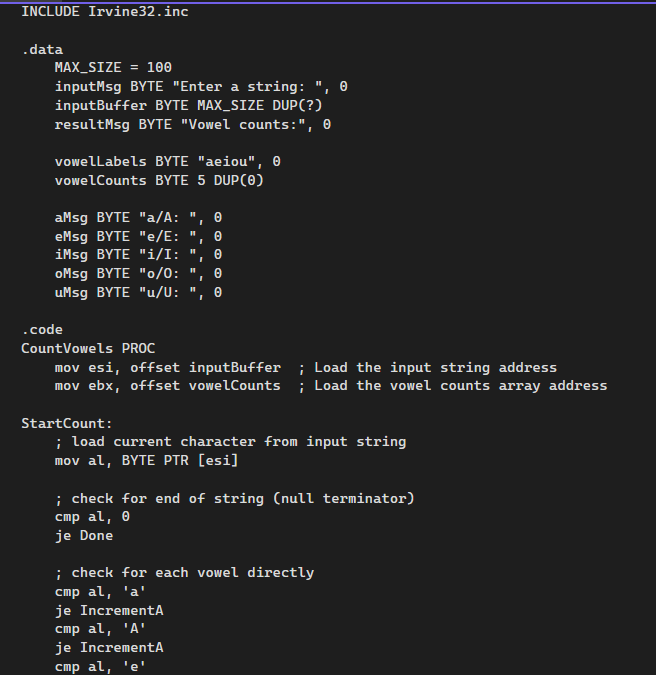
call crlf

call PrintVowelCounts

exit

main ENDP

END main



A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Question #05**

INCLUDE Irvine32.inc

.DATA

msg1 BYTE "Test case ", 0

msg2 BYTE " - Result: ", 0

newline BYTE 13, 10, 0

.CODE

DifferentInputs PROC, val1:DWORD, val2:DWORD, val3:DWORD

mov eax, 1

; compare first and second values

mov ecx, val1

cmp ecx, val2

je NotDifferent

; compare first and third values

cmp ecx, val3

je NotDifferent

; compare second and third values

mov ecx, val2

cmp ecx, val3

je NotDifferent

ret

NotDifferent:

mov eax, 0

ret

DifferentInputs ENDP

main PROC

; Test case 1: All different values

mov edx, OFFSET msg1

call WriteString

mov eax, 1 ; Values to print

call WriteDec

mov edx, OFFSET msg2

call WriteString

INVOKE DifferentInputs, 1, 2, 3

call WriteDec

call Crlf

; Test case 2: Two values are the same

mov edx, OFFSET msg1

call WriteString

mov eax, 2

call WriteDec

mov edx, OFFSET msg2

call WriteString

INVOKE DifferentInputs, 5, 5, 7

call WriteDec

call Crlf

; Test case 3: All values are the same

mov edx, OFFSET msg1

call WriteString

mov eax, 3

call WriteDec

mov edx, OFFSET msg2

call WriteString

INVOKE DifferentInputs, 4, 4, 4

call WriteDec

call Crlf

; Test case 4: Different values

mov edx, OFFSET msg1

call WriteString

mov eax, 4

call WriteDec

mov edx, OFFSET msg2

call WriteString

INVOKE DifferentInputs, 10, 20, 30

call WriteDec

call Crlf

; Test case 5: Negative values

mov edx, OFFSET msg1

call WriteString

mov eax, 5

call WriteDec

mov edx, OFFSET msg2

call WriteString

INVOKE DifferentInputs, -1, -2, -3

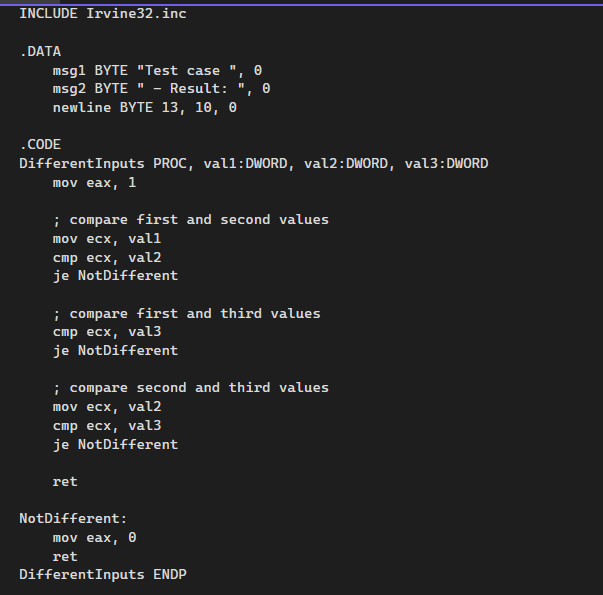
call WriteDec

call Crlf

exit

main ENDP

END main



A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A black background with white text

Description automatically generated

**Question #06**

INCLUDE Irvine32.inc

.data

inputStr BYTE "###ABC", 0

removeChar BYTE "#", 0

resultMsg BYTE "Resulting string: ", 0

.code

RemoveLeadingChar PROC

; esi -> pointer to string

; edi -> character to remove

push esi

push edi

; load the address of the string into esi and the character to remove into edi

mov esi, offset inputStr

mov al, [esi]

mov bl, [removeChar]

RemoveLoop:

; compare current character with the one to remove, if they don't match, stop.

cmp al, bl

jne EndRemove

inc esi

mov al, [esi]

jmp RemoveLoop

EndRemove:

mov edi, offset inputStr

CopyLoop:

mov al, [esi]

cmp al, 0

je EndCopy

mov [edi], al

inc esi

inc edi

jmp CopyLoop

EndCopy:

mov byte ptr [edi], 0

pop edi

pop esi

ret

RemoveLeadingChar ENDP

main PROC

mov edx, offset inputStr

call WriteString

call crlf

INVOKE RemoveLeadingChar

mov edx, offset resultMsg

call WriteString

mov edx, offset inputStr

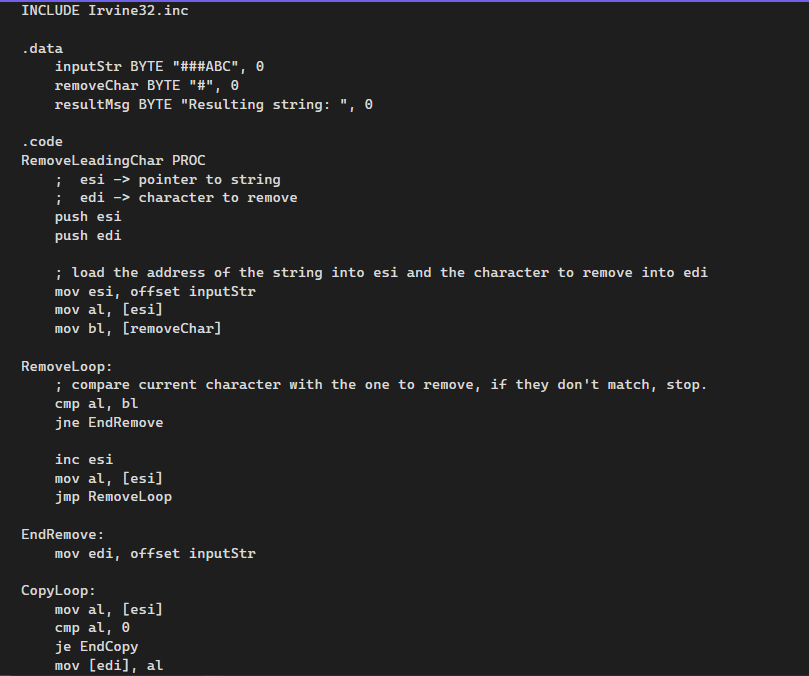
call WriteString

call crlf

exit

main ENDP

END main



A screenshot of a computer program

Description automatically generated A black background with white text

Description automatically generated