Q1

Saturday, November 30, 2024 11:34 PM

Q1

```
divide  proc    ; dividend in eax , divisor in edx
        cmp     eax, 5h
        jg      endd
        mov     edx,0
        div     ebx
        call    divide
endd:
        ret
divide  endp
```

Q2

```
include irvine32.inc

.data
    array    dword    20 dup(?)
    found msg    byte " found: ",0
    not-found msg  byte "not found",0
    arr Size   dword   ?
.code
    call   readint
    mov    ebx, eax
    mov    esi, offset arr
    mov    edx, length of arr
    mov    arr Size, edx
    mov esi, offset arr

  invoke    findval , esi, ecx, arrSize, ebx

  exit
  main endp

findval proc uses  eax, esi, ecx, edx, ebx

    cmp   ecx, edx
    jl   notfound

    mov eax, [esi + ecx *4]
    cmp   eax, ebx
    je  found
    inc  eax
    invoke   findval , esi, ecx, edx, ebx
  ret
  found:
      mov  edx, offset foundmsg
      call  write string
      mov  eax, ecx
      call  write dec
      ret

  not found:
      mov   edx, offset not-found msg
      call  write string
      ret
findval   endp

end main
```

Q3

```
include irvine32.inc

.data
    source byte "This is a source string",0
    something byte 256 dup(?)
    strlen dword ?
    ind dword 0

.code
main proc
    mov edx, offset source
    call str_length

    mov strlen, eax
    mov edi, offset something
    mov esi, offset source
    mov ecx, strlen

nextchars:
    mov al, byte ptr [esi]
    test al, 01
    je done

    mov ebx, offset target
    mov edx, ind
    mov ah, al
    mov ecx, edx
    cld
    repe scasb
    jz skipchars

    mov byte ptr [esi], al
    inc edi
    inc ind

skipchars:
    inc esi
    loop nextchars

done:
    mov byte ptr [edi], 0

mov edx, offset target
call writestring

exit
main endp

end main
```

Q4

```
.data
    prompt  byte  "Enter a string",0
    cha     byte  "a/A",0
    che     byte  "e/E",0
    chi     byte  "i/I",0
    cho     byte  "o/O",0
    chu     byte  "u/U",0
    countA  dword 0
    countE  dword 0
    ; count for i,o,u.
    strInput byte 50 dup(?).
.code
main Proc

    mov edx, offset prompt
    call writestring
    mov edx, offset strInput
    call readstring
    mov esi, offset strInput
  nextchar:
    mov al, byte ptr [esi]
    cmp al,0
    jz displaycount

    cmp al, 'a'
    je inc_A
    cmp al, 'A'
    je inc_A
    cmp al, 'e'.
    je inc_E
    cmp al, 'E'
    je inc_E
    cmp al, 'i'.
    je inc_I
    cmp al, 'I'
    je inc_I

    cmp al, 'o'.
    je inc_O
    cmp al, 'O'
    je inc_O

    cmp al, 'u'.
    je inc_U
    cmp al, 'U'
    je inc_U
    inc esi
    jmp nextChar

inc_A:
    inc countA
    inc esi
    jmp nextchecc

inc_E:
    inc countE
    inc esi
    jmp nextchecc

inc_I:
    inc countI
    inc esi
    jmp nextchecc


inc_O:
    inc countO
    inc esi
    jmp nextchecc


inc_U:
    inc countU
    inc esi
    jmp nextchecc

displaycount:
    mov eax, countA
    mov edx, offset cha
    call writestring
    call crlf

    mov eax, countE
    mov edx, offset che
    call writestring
    call crlf

    mov eax, countI
    mov edx, offset chi
    call writestring
    call crlf

    mov eax, countO
    mov edx, offset cho
    call writestring
    call crlf

    mov eax, countU
    mov edx, offset chu
    call writestring
    call crlf

exit
main Endp
end main
```

Q5

```
include irvine32.inc
.data
    msg     byte "diff values",0
    msg1    byte "same values",0

.code
    diffInputs  proto : dword, :dword, :dword

main proc

    mov  eax, 1
    mov  ebx, 2
    mov  ecx, 3
    call diffInputs

    mov  edx, offset msg
    call writestring
    call writedec


    mov  eax, 1
    mov  ebx, 1
    mov  ecx, 1
    call diffInputs

    mov  edx, offset msg1
    call writestring
    call writedec

exit
main endp

diffInput proc : dword, :dword, :dword
    cmp  eax, ebx
    jc   Same val

    cmp  ebx, eax
    je   same val

    cmp  eax, ecx
    je   saneval
    mov  eax, 1
    ret 1

Samevalue:
    mov  eax,0
    ret

diffInput endp

end main
```

q6

Sunday, December 1, 2024 10:47 AM

Q6

```
include irvine32.inc

.data
    str1 byte "###ABC",0

.code
    main proc

    l1:
        mov al, [edx]
        cmp al,0
        je done

        cmp al, [esp+4]
        jne done
        inc edx
        loop l1

    done:
        mov ecx, edx
        mov esi, offset str1
        mov edi, ecx
        cld
        rep movsb
        mov edx, offset str1
        call writestring

    exit
    main endp
    end main
```