23k-2001
BCS-3J
Date ____ 20__

COAL Assignment: 03

ANSWER#01:

```
INCLUDE Irvine32.inc
.code
main PROC
    mov eax, 0D4A4h      ; dividend
    mov ebx, 0Ah         ; divisor
    call divideProc      ; our recursive function
    call writehex
exit
main ENDP

divideProc PROC
    cmp eax, 5h
    jle done
    mov edx, 0
    div ebx
    call divideProc

    done:
    ret
divideProc ENDP

END main
```

Date ____ 20__

ANSWER#02:

```
INCLUDE Irvine32.inc
.data
    arr DWORD 50 DUP(?)
    yes BYTE "Number found at index: ", 0
    no BYTE "Number not found in the array. ", 0
    input BYTE "Input a number to search: ", 0
    findNum PROTO p1: PTR DWORD, p2: DWORD, p3: DWORD, p4: DWORD
.code
main PROC
    mov edx, offset input
    call writestring
    call readint

    mov ebx, eax
    mov ecx, 0
    mov esi, offset arr
    call crlf
    INVOKE findNum, esi, ecx, lengthof arr, ebx
exit
main ENDP

findNum PROC uses eax esi ecx edx ebx, p1: PTR DWORD, p2: DWORD,
                                       p3: DWORD, p4: DWORD

    mov esi, p1
    mov ecx, p2
    cmp ecx, p3
    jge notfound
    mov eax, [esi+ecx*4]
    cmp eax, p4
    je found
    inc ecx
    INVOKE findNum, esi, ecx, p3, p4
    ret

found:
    mov edx, offset yes
    call writestring
    mov eax, ecx
    call writedec
    jmp done
notfound:
    mov edx, offset no
    call writestring
    jmp done
done:
    ret
findNum ENDP

END main
```

(example for Q2 : stack values) Date____20__

ebx: contains number to search.

let: arr DWORD 1,2,3,4,5 , offset arr: 1000h

inputted num: 5 → (ebx)

p1: arr offset, p2: ecx (inc in each call)

p3: lengthof arr , p4: ebx (5)

First call:

ret address to main	
p1 = 1000h	
p2 = 0	
p3 = 10	
p4 = 5	(notfound)

Second call:

ret address to main	
p1 = 1000h	
p2 = 0	
p3 = 10	
p4 = 5	
ret to first call	
p1 = 1000h	
p2 = 1	
p3 = 10	
p4 = 5	(notfound)

Third call:

ret address to main	
p1 = 1000h	
p2 = 0	
p3 = 10	
p4 = 5	
ret to first call	
p1 = 1000h	
p2 = 1	
p3 = 10	
p4 = 5	
ret to second call	
p1 = 1000h	
p2 = 2	
p3 = 10	
p4 = 5	(notfound)

Date____20__

Fourth call:

ret address to main	
p1 = 1000h	
p2 = 0	
p3 = 10	
p4 = 5	
ret to first call	
p1 = 1000h	
p2 = 1	
p3 = 10	
p4 = 5	
ret to second call	
p1 = 1000h	
p2 = 2	
p3 = 10	
p4 = 5	
ret to third call	
p1 = 1000h	
p2 = 3	
p3 = 10	
p4 = 5	(notfound)

Fifth call:

ret to address to main	
p1 = 1000h	
p2 = 0	
p3 = 10	
p4 = 5	
ret to first call	
p1 = 1000h	
p2 = 1	
p3 = 10	
p4 = 5	
ret to second call	
p1 = 1000h	
p2 = 2	
p3 = 10	
p4 = 5	
ret to third call	
p1 = 1000h	
p2 = 3	
p3 = 10	
p4 = 5	
ret to fourth call	
p1 = 1000h	
p2 = 4	
p3 = 10	
p4 = 5	
ret	(found)

eax=

eax=

eax=

eax=4

Final output: (eax = 4)

⇒ Number found at index: 4

ANSWER#03:

```
INCLUDE Irvine32.inc
.data
  source byte "This is the source string",0
  target byte 100 DUP(0)

.code
main PROC
    push offset source
    push lengthof source
    push offset target
    call copySingle

    mov edx, offset target
    call writestring
    call crlf
exit
main ENDP

copySingle PROC
    push ebp
    mov ebp, esp
    mov esi, [ebp+16]
    test esi, 1
    mov ebx, 0
    mov ecx, 0
    L1:
     cld
     lodsb
     push ecx
     mov edi, [ebp+8]
     repnz scasb
     pop ecx
     jz skip

     stosb
     inc ecx

     skip:
       inc ebx
     cmp ebx, [ebp+12]
     jnz L1

     pop ebp
     ret 12
copySingle ENDP

END main
```

Date

ANSWER#04:

```
INCLUDE Irvine32.inc
.data
    cA byte "A|a: ",0
    cE byte "E|e: ",0
    cI byte "I|i: ",0
    cO byte "O|o: ",0
    cU byte "U|u: ",0
    countA Dword 0
    countE Dword 0
    countI Dword 0
    countO Dword 0
    countU Dword 0
    inputstr byte "Advanced Programming in UNIX Environment",0
```

pg.01

```
.code
main PROC
    lea edx, inputStr
    mov ecx, lengthof inputStr
    lea esi, inputStr
calculate:
    lodsb
    cmp al,0
    je done

    ; for 'A' or 'a'
    cmp al, 'A'
    je incrementA
    cmp al, 'a'
    je incrementA

    ; for 'E' or 'e'
    cmp al, 'E'
    je incrementE
    cmp al, 'e'
    je incrementE

    ; for 'I' or 'i'
    cmp al, 'I'
    je incrementI
    cmp al, 'i'
    je incrementI

    ; for 'O' or 'o'
    cmp al, 'O'
    je incrementO
    cmp al, 'o'
    je incrementO
```

pg.02

```
    ; for 'U' or 'u'
    cmp al, 'U'
    je incrementU
    cmp al, 'u'
    je incrementU

    jmp calculate

incrementA:
    inc countA
    jmp calculate

incrementE:
    inc countE
    jmp calculate

incrementI:
    inc countI
    jmp calculate

incrementO:
    inc countO
    jmp calculate

done:
    lea edx, cA
    call writestring
    mov eax, countA
    call writedec
    call crlf
```

pg.03

```
    lea edx, cE
    call writestring
    mov eax, countE
    call writedec
    call crlf

    lea edx, cI
    call writestring
    mov eax, countI
    call writedec
    call crlf

    lea edx, cO
    call writestring
    mov eax, countO
    call writedec
    call crlf

    lea edx, cU
    call writestring
    mov eax, countU
    call writedec
    call crlf

    exit
main ENDP
END main
```

Dated:

ANSWER#05 :

```
INCLUDE Irvine32.inc
.data
   differentInputs PROTO, var1:dword, val2:dword, var3:dword
   res byte "EAX= ", 0
.code
main PROC
   mov edx, offset res
   INVOKE differentInputs, 1, 2, 3
   call writestring
   call writedec
   call crlf

   INVOKE differentInputs, 4, 5, 6
   call writestring
   call writedec
   call crlf

   INVOKE differentInputs, 7, 8, 9
   call writestring
   call writedec
   call crlf

   INVOKE differentInputs, 10, 11, 12
   call writestring
   call writedec
   call crlf

   INVOKE differentInputs, 1, 1, 0
   call writestring
   call writedec
   call crlf
```

Dated:

```
   exit
main ENDP

differentInputs PROC val1:dword, val2:dword, val3:dword
   mov eax, val1
   mov ebx, val2
   mov ecx, val3

   cmp eax, ebx
   je same
   cmp eax, ecx
   je same
   cmp ebx, ecx
   je same

   mov eax, 1
   ret
   same:
      mov eax, 0
      ret
differentInputs ENDP
END main
```

Dated: **ANSWER#06:** 23K-2001

```
INCLUDE Irvine32.inc
.data
input1 byte "Input your string: ",0
input2 byte "Input leading character: ",0
source byte 50 DUP(?)
key byte ?
.code
main PROC
    mov edx, offset input1
    call writestring
    mov edx, offset source
    mov ecx, sizeof source
    call readstring
    call crlf
    mov edx, offset input2
    call writestring
    call readchar
    call writechar
    mov key, al

    push offset source
    push sizeof source
    movzx eax, key
    push eax
    call modifiedStrtrim

    call crlf
    mov edx, offset source
    call writestring
    exit
main ENDP
```

pg. 1

pg. 2

```
modifiedStrtrim PROC
    push ebp
    mov ebp, esp
    mov edi, [ebp+16]
    mov al, [ebp+8]
    mov ecx, [ebp+12]
    cld
    repz scasb
    mov esi, edi
    dec esi
    mov edi, [ebp+16]
    L1:
      lodsb
      stosb
      cmp al, 0
      jnz L1
    pop ebp
    ret 8
modifiedStrtrim ENDP

END main
```