**K18-0208 SECTION: H**

**QUESTION: 1**

include C:\Irvine\Irvine32.inc

includelib C:\Irvine\Irvine32.Lib

includelib C:\Irvine\Kernel32.Lib

includelib C:\Irvine\User32.Lib

.data

dividend sbyte -20

.code

main proc

mov eax,0

mov al, dividend

mov bl,+2

idiv bl

call dumpregs

movsx eax,al

call writeInt

call crlf

mov eax,0

mov al,dividend

cbw

mov bl,+2

idiv bl

call dumpregs

movsx eax,al

call writeInt

exit

main endp

end main

**QUESTION: 2**

include C:\Irvine\Irvine32.inc

includelib C:\Irvine\Irvine32.Lib

includelib C:\Irvine\Kernel32.Lib

includelib C:\Irvine\User32.Lib

.data

array1 SWORD 20 DUP(?)

result sword 0

str1 BYTE "SUM OF RANDOM ODD NUMBERS ARE: ", 0

str2 BYTE "RANDOM NUMBER: ", 0

.code

main proc

mov ebx, offset array1

mov ecx, 20

call randomize

fillArray:

mov eax , 99

call randomrange

sub eax, 50

mov [ebx] , eax

add ebx, 2

loop fillArray

mov ecx, lengthof array1

mov esi,0

print:

movsx eax, [array1[esi]]

add esi, 2

mov edx, offset str2

call writestring

call writeint

call crlf

loop print

mov ecx, lengthof array1

mov eax, 0

mov ebx, 0

mov edx, 0

mov esi, 0

mov edi, 0

mov bl, 2

call oddSum

mov edx, OFFSET str1

call crlf

call writestring

call writeint

call crlf

exit

main endp

oddSum proc

push ebp

mov ebp, esp

cmp ecx, 0

jz  return

jg  check

check:

mov ax, [array1[esi]]

cmp ax, 49

jg range

cmp ax, -50

jl range

idiv bl

cmp ah, 0

jne getOdd

range:

dec ecx

add esi, 2

call oddSum

jmp return

getOdd:

mov ax, [array1[esi]]

add result, ax

add esi, 2

dec ecx

call oddSum

return:

mov eax, 0

movsx eax, result

pop ebp

ret

oddSum endp

end main

**QUESTION: 3**

include C:\Irvine\Irvine32.inc

includelib C:\Irvine\Irvine32.Lib

includelib C:\Irvine\Kernel32.Lib

includelib C:\Irvine\User32.Lib

.data

temp1 DWORD 1

temp2 DWORD 1

multiplier DWORD 2

space DWORD 32

.code

main proc

mov edx, 0

mov ecx, 5 ; 5 levels

mov esi,1

call makePattern

call crlf

exit

main endp

makePattern proc

push ebp

mov ebp, esp

cmp ecx, 0

je return

mov ebx, ecx

mov eax, space

printSpaces:

call writechar

loop printSpaces

mov ecx, temp2

printInt:

mov eax, temp1

call writedec

mul multiplier

mov temp1, eax

mov eax, space

call writechar

loop printInt

mov ecx, ebx

dec ecx

add temp2, 2 ; odd iterations

call crlf

call makePattern

jmp return

return:

pop ebp

ret

makePattern endp

end main

**QUESTION: 4**

include C:\Irvine\Irvine32.inc

includelib C:\Irvine\Irvine32.Lib

includelib C:\Irvine\Kernel32.Lib

includelib C:\Irvine\User32.Lib

.data

stringToCopy BYTE "SOURCE: ",0

stringCopied BYTE "DESTINATION: ",0

equalString BYTE "STRINGS ARE EQUAL", 0

unEqualString BYTE "STRINGS ARE UNEQUAL", 0

notFound BYTE "STRING NOT FOUND", 0

found BYTE "STRING FOUND AT ", 0

subs BYTE "SUBSTRING: ", 0

stringCmp1 BYTE "JOHN",0

stringCmp2 BYTE 20 DUP(?),0

subString1 BYTE 20 DUP(?),0

findstring1 BYTE "HN",0

firstIndex DWORD 0

secondIndex DWORD 3

.code

main PROC

mov ecx, lengthof stringCmp1

push OFFSET stringCmp2

push OFFSET stringCmp1

call stringCopy

add ebp, 8

mov ecx, lengthof stringCmp1

push OFFSET stringCmp2

push OFFSET stringCmp1

call stringCompare

add ebp, 8

push OFFSET stringCmp1

push OFFSET findString1

call findString

add ebp, 8

push firstIndex

push secondIndex

push OFFSET stringCmp1

call subString

exit

main ENDP

stringCopy PROC

push ebp

mov ebp, esp

mov esi, [ebp+8]

mov edi, [ebp+12]

cld

rep movsb

mov edx, OFFSET stringToCopy

call writestring

mov edx, [ebp+8]

call writestring

call crlf

mov edx, OFFSET stringCopied

call writestring

mov edx, [ebp+12]

call writestring

call crlf

pop ebp

ret

stringCopy ENDP

stringCompare PROC

push ebp

mov ebp, esp

mov esi, [ebp+8]

mov edi, [ebp+12]

cld

repe cmpsb

jne notEqual

equal:

mov edx, OFFSET equalString

call writestring

call crlf

jmp return

notEqual:

mov eax, OFFSET unEqualString

call writestring

call crlf

jmp return

return:

pop ebp

ret

stringCompare ENDP

findString PROC

push ebp

mov ebp, esp

mov esi, [ebp+12]

mov edi, [ebp+8]

mov ebx, 0

mov edx, 0

mov eax, 0

mov ecx, LENGTHOF stringCmp1

find:

mov edx, ecx

mov ecx, LENGTHOF findString1

dec ecx

repe cmpsb

je equal

inc eax

mov edi, OFFSET findString1

mov ecx, edx

loop find

mov edx , OFFSET notFound

call writestring

call crlf

pop ebp

ret

equal:

mov edx , OFFSET found

call writestring

call writedec

call crlf

pop ebp

ret

findString ENDP

subString PROC

push ebp

mov ebp, esp

mov ebx, 0

mov eax, 0

mov esi, [ebp+8]

mov ebx, [ebp+12]

mov eax, [ebp+16]

add esi, eax

sub ebx, eax

mov ecx, ebx

mov edi, OFFSET subString1

rep movsb

mov edx, OFFSET subs

call writestring

mov edx, OFFSET subString1

call writestring

call crlf

pop ebp

ret

subString ENDP

end main

**QUESTION: 5**

include C:\Irvine\Irvine32.inc

includelib C:\Irvine\Irvine32.Lib

includelib C:\Irvine\Kernel32.Lib

includelib C:\Irvine\User32.Lib

.data

numbers byte 24 dup(?)

rs = 6

prompt1 byte "rows",0

prompt2 byte "col ",0

rows byte 6

col = 4

sum word 0

prompt3 byte "table",0

prompt4 byte "enter numbers ", 0

cols byte 4

divisor byte 24

.code

main proc

mov ebx , offset numbers

mov edi , offset numbers

mov ecx , 24

mov edx , offset prompt4

call crlf

call writestring

call crlf

mov edx , 0

L1:

call readdec

mov [edi] , al

inc edi

loop L1

mov edi , 0

mov esi , offset numbers

mov ecx , 24

call crlf

mov esi , 0

mov ecx , col

mov edx , offset prompt1

call writestring

call crlf

avgr:

mov esi , 0

mov edi , ecx

mov ecx , rs

sumr:

add al , [ebx+esi]

inc esi

loop sumr

add sum , ax

div rows

mov ah , 0

mov ecx , edi

call writedec

call crlf

mov eax,0

add ebx , rs

loop avgr

call crlf

mov edx , offset prompt2

call writestring

call crlf

mov ebx , offset numbers

mov eax , 0

mov ecx , rs

avgc:

mov edi , ecx

mov eax , 0

mov esi , ebx

mov cl , 4

sums:

add al , [esi]

add esi , 6

loop sums

div cols

call writedec

call crlf

inc ebx

mov ecx , edi

loop avgc

mov edx , offset prompt3

call writestring

call crlf

call crlf

mov eax , 0

movzx eax , sum

div divisor

mov ah , 0

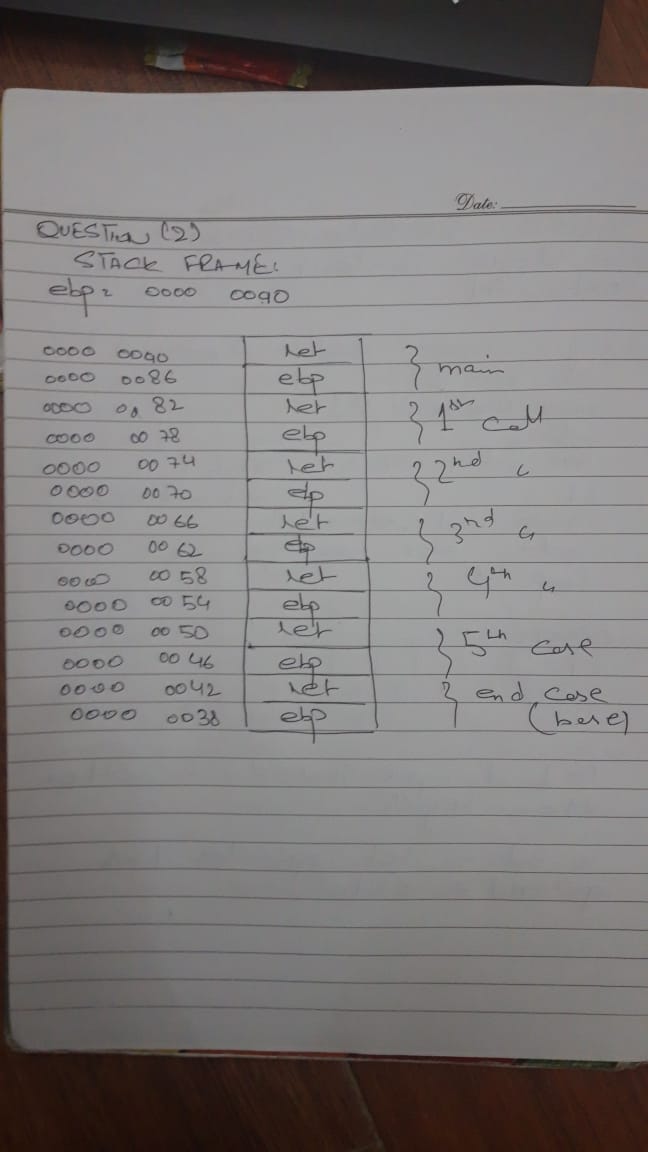
dec eax

call writedec

exit

main endp

end main

**Stack frame (QUESTION: 2)**