COAL Lab 10 Assignment:

# Name: Owais Ali Khan

# Section: 3-F

# Roll no: 21K-3298

Question # 01:

TITLE Question 1

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

array DWORD 5,1,7,2,9,3,6,8

count = ($ - array) / TYPE array

.code

main PROC

push OFFSET array

push count

mWrite "Unsorted array: "

call print ; Before Sorting

push OFFSET array

push count

call BubbleSort ; Bubble Sorting

push OFFSET array

push count

mWrite "BubbleSorted array: "

call print ; After Sorting

exit

main ENDP

BubbleSort PROC

push ebp

mov ebp, esp

mov ecx, [ebp + 8]

mov esi, [ebp + 12]

L1:

push esi

push ecx

mov ecx, [ebp + 8]

dec ecx

L2:

mov eax, [esi]

cmp eax, [esi + 4]

jbe NotSwapped

xchg eax, [esi + 4]

mov [esi], eax

NotSwapped:

add esi, 4

loop L2

pop ecx

pop esi

loop L1

mov esp, ebp

pop ebp

ret 8

BubbleSort ENDP

print PROC

push ebp

mov ebp, esp

mov ecx, [ebp + 8]

mov esi, [ebp + 12]

L2:

mov eax, [esi]

call WriteDec

mWrite " "

add esi, TYPE array

loop L2

call Crlf

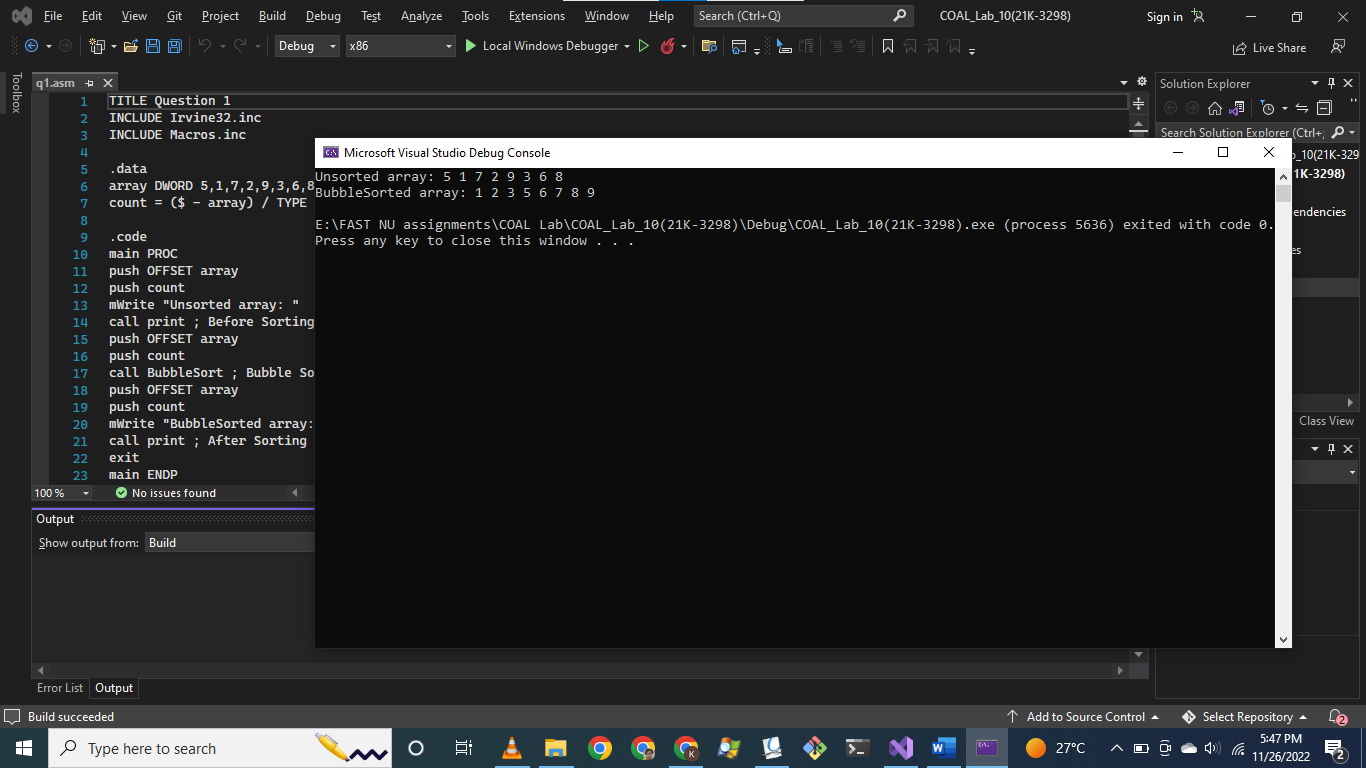
mov esp, ebp

pop ebp

ret 8

print ENDP

END main



Question # 02:

TITLE Question 2

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

count DWORD 0

sum DWORD 0

value DWORD ?

divisor DWORD 10

.code

main PROC

call TakeInput

exit

main ENDP

TakeInput PROC

mWrite "Procedure: TakeInput"

call Crlf

mWrite "ESP = "

push eax

mov eax, esp

call WriteHex

call Crlf

call Crlf

pop eax

push ebp

mov ebp, esp

mWrite "Enter number to check for Armstrong: "

call ReadDec

mov value, eax

mov edx, 0

L1:

mov edx, 0

cmp eax, 0

je L2

add count, 1

div divisor

jmp L1

call Crlf

L2:

mWrite "Count = "

mov eax, count

call WriteDec

call Crlf

call Crlf

push value

push count

call Armstrong

mov esp, ebp

pop ebp

ret

TakeInput ENDP

Armstrong PROC

call Crlf

mWrite "Procedure: Armstrong"

call Crlf

mWrite "ESP = "

push eax

mov eax, esp

call WriteHex

call Crlf

call Crlf

pop eax

push ebp

mov ebp, esp

mov ecx, [ebp + 8]

mov eax, [ebp + 12]

mov edx, 0

L3:

mov edx, 0

div divisor

xchg eax, edx

push ecx

push edx

push eax

mov ebx, eax

mov ecx, [ebp + 8]

mov edx, 0

mov eax, 1

mWrite "Powers of "

L4:

mul ebx

call WriteDec

mWrite " "

loop L4

call Crlf

add sum, eax

pop eax

pop edx

pop ecx

xchg eax, edx

loop L3

mWrite "Sum = "

mov eax, sum

call WriteDec

call Crlf

call Crlf

push eax ; Calculated sum for Armstrong

push [ebp + 12] ; Original input value

call Display

mov esp, ebp

pop ebp

ret 8

Armstrong ENDP

Display PROC

call Crlf

mWrite "Procedure: Display"

call Crlf

mWrite "ESP = "

push eax

mov eax, esp

call WriteHex

call Crlf

call Crlf

pop eax

push ebp

mov ebp, esp

mov eax, [ebp + 8]

mov ebx, [ebp + 12]

cmp eax, ebx

jne NotArmstrong

mWrite "Given number is Armstrong Number."

call Crlf

jmp ENDD

NotArmstrong:

mWrite "Given number is not Armstrong Number."

call Crlf

ENDD:

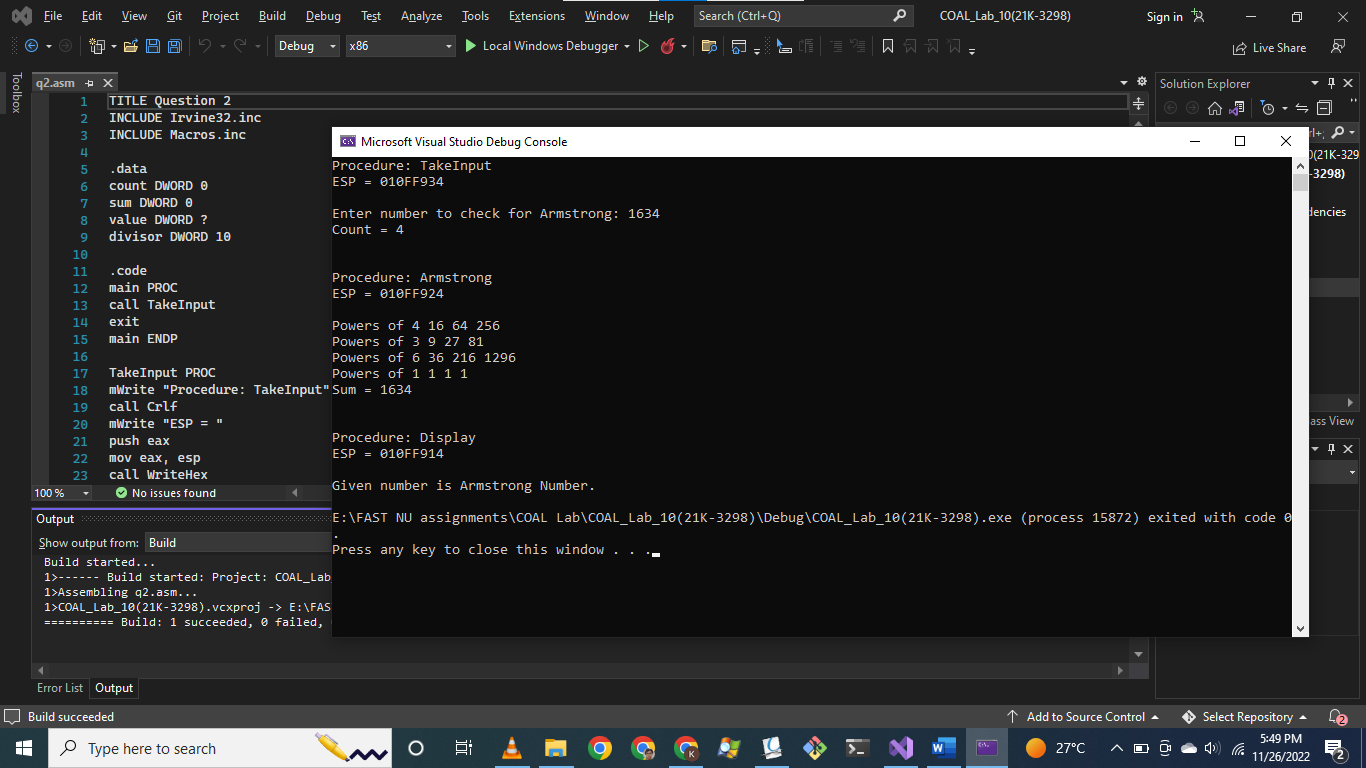
mov esp, ebp

pop ebp

ret 8

Display ENDP

END main



Question # 03:

TITLE Question 3

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

string BYTE "Owais Ali Khan", 0

.code

main PROC

mov eax, 0

mov esi, 0

mov edi, 0

mWrite "Original string: "

mov edx, OFFSET string

call WriteString

call Crlf

push OFFSET string

call Reverse

mWrite "Reversed string: "

mov edx, OFFSET string

call WriteString

call Crlf

exit

main ENDP

Reverse PROC

push ebp

mov ebp, esp

mov esi, [ebp + 8]

movzx eax, BYTE PTR [esi]

cmp eax, 0

jne RECURSE

mov esp, ebp

pop ebp

ret 4

RECURSE:

push eax

add esi, 1

push esi

call Reverse

pop eax

mov string[edi], al

add edi, 1

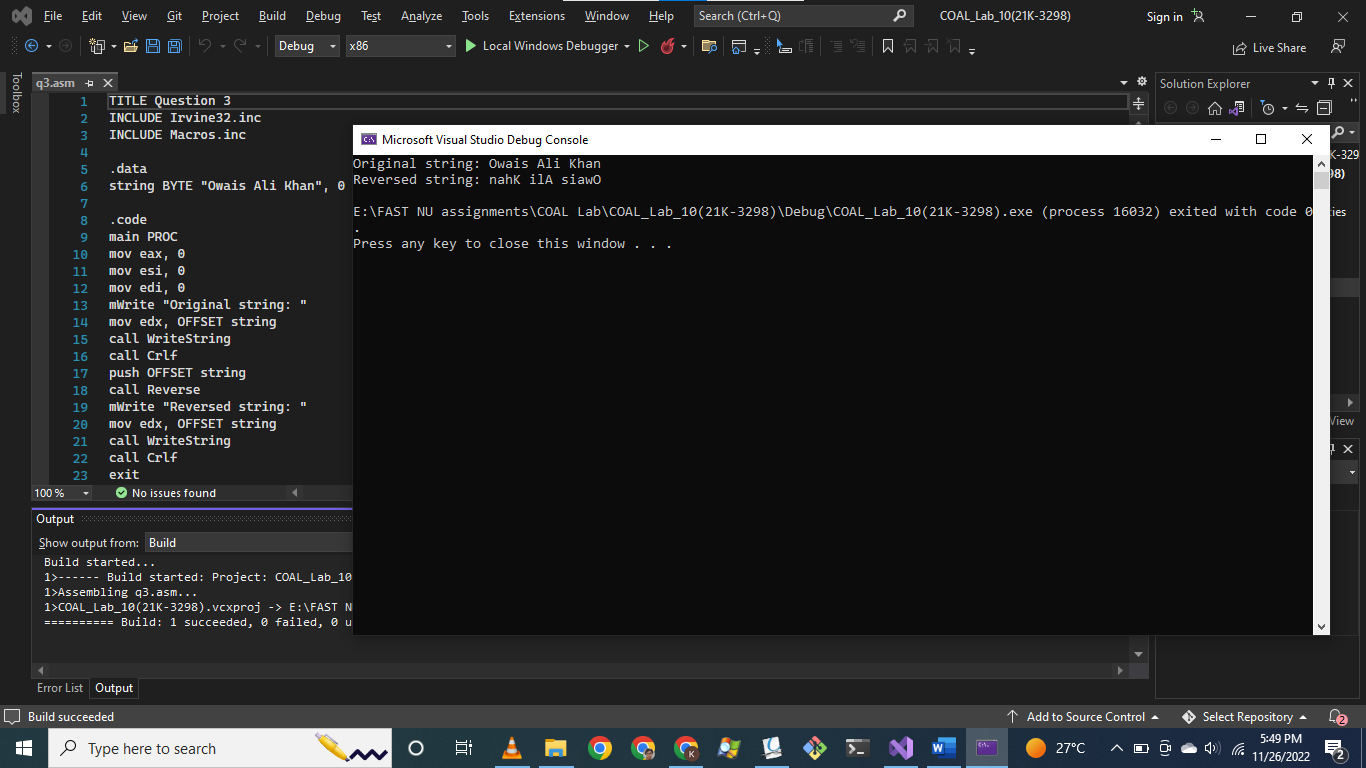
mov esp, ebp

pop ebp

ret 4

Reverse ENDP

END main



Question # 04:

TITLE Question 4

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

.code

main PROC

call LocalSquare

exit

main ENDP

LocalSquare PROC

ENTER 4, 0

mWrite "Enter value: "

call ReadInt

call Crlf

mWrite "Original value: "

call WriteInt

call Crlf

mov [ebp - 4], eax

imul eax, [ebp - 4]

mWrite "Squared value: "

call WriteDec

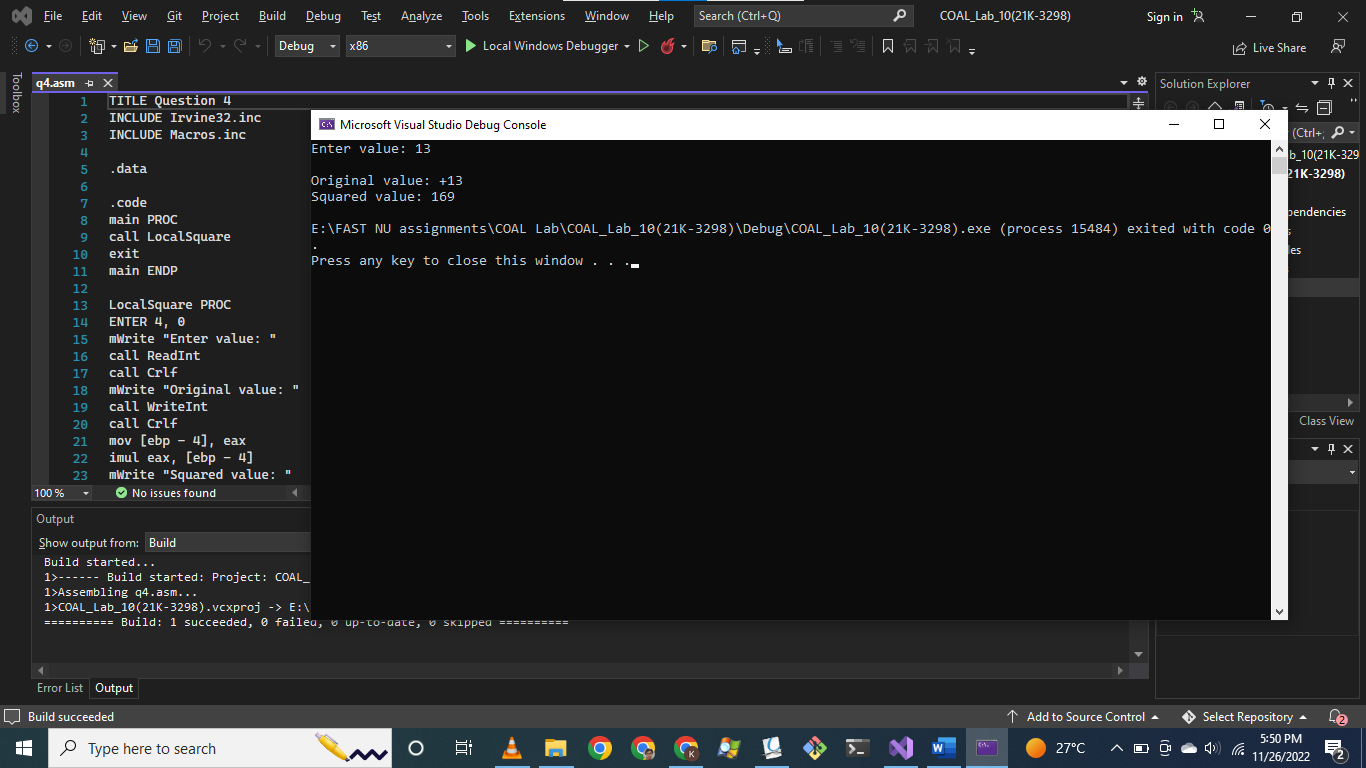
call Crlf

LEAVE

ret

LocalSquare ENDP

END main



Question # 05:

TITLE Question 5

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

n DWORD ?

.code

main PROC

mWrite "Enter number to calculate factorial: "

AGAIN:

call ReadInt

cmp eax, 0

jg CONTINUE

mWrite "Enter number greater than 0: "

jmp AGAIN

CONTINUE:

call Crlf

mWrite "Original = "

call WriteDec

call Crlf

mov n, eax

call Fact

mov eax, n

mWrite "Factorial = "

call WriteDec

call Crlf

exit

main ENDP

Fact PROC

push ebp

mov ebp, esp

cmp eax, 1

jg RECURSE

mov esp, ebp

pop ebp

ret

RECURSE:

mov edx, 0

dec eax

push eax

call Fact

mov eax, n

pop ebx

mul ebx

mov n, eax

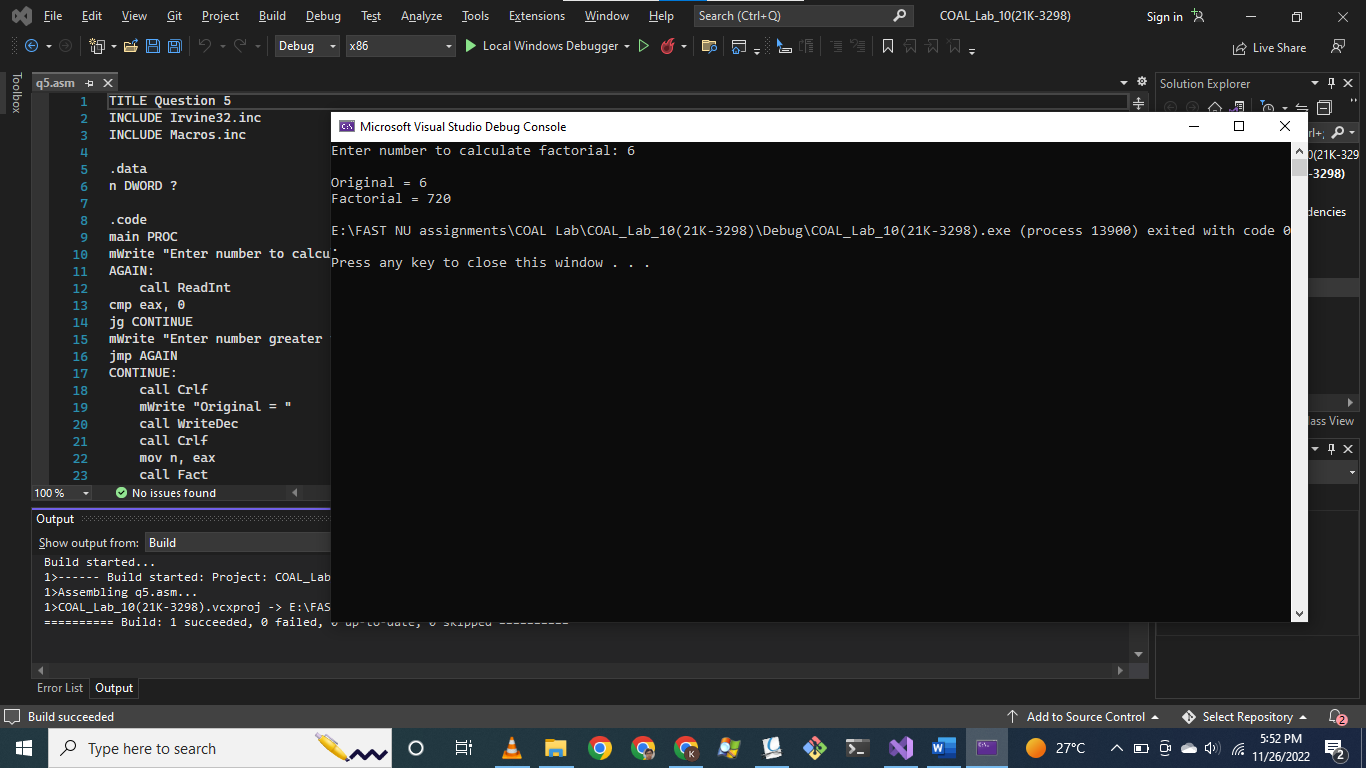
mov esp, ebp

pop ebp

ret

Fact ENDP

END main



Question # 06:

TITLE Question 6

INCLUDE Irvine32.inc

INCLUDE Macros.inc

.data

numbers DWORD 0,0,0,0

count = ($ - numbers) / TYPE numbers

.code

main PROC

mov eax, 0

mov esi, 0

mov ecx, count

INPUT:

mWrite "Enter number: "

call ReadDec

mov numbers[esi], eax

add esi, TYPE numbers

loop INPUT

push DWORD PTR [numbers]

call CheckPrime

jnc NotPrime

push DWORD PTR [numbers + 4]

call CheckPrime

jnc NotPrime

push DWORD PTR [numbers + 8]

call CheckPrime

jnc NotPrime

push DWORD PTR [numbers + 12]

call CheckPrime

jnc NotPrime

call Crlf

mWrite "All numbers are prime."

call Crlf

push OFFSET numbers

call LargestPrime

jmp ENDD

NotPRIME:

call Crlf

mWrite "All numbers are not prime."

ENDD:

call Crlf

exit

main ENDP

CheckPrime PROC ; Returns bool value in Carry Flag (1 for prime, 0 for non-prime)

push ebp

mov ebp, esp

mov eax, [ebp + 8]

cmp eax, 2

jl PRIME\_FALSE

mov ecx, eax

sub ecx, 1

Check:

push eax

mov edx, 0

cmp ecx, 1

jle PRIME\_TRUE

div ecx

cmp edx, 0

je PRIME\_FALSE

pop eax

loop Check

PRIME\_FALSE:

clc

pushfd

jmp ENDD

PRIME\_TRUE:

stc

pushfd

ENDD:

popfd

mov esp, ebp

pop ebp

ret 4

CheckPrime ENDP

LargestPrime PROC

push ebp

mov ebp, esp

mov esi, [ebp + 8]

mov eax, [esi]

mov ecx, count - 1

Iterate:

add esi, 4

cmp eax, [esi]

jge CONTINUE

mov eax, [esi]

CONTINUE:

loop Iterate

mWrite "Largest prime is: "

call WriteDec

mov esp, ebp

pop ebp

ret 4

LargestPrime ENDP

END main

