**COAL Lab # 09**

## **Question 1:**

INCLUDE Irvine32.inc

.data

x DWORD 2

y DWORD 3

z DWORD 4

.code

ThreeProd PROC

push ebp

mov ebp, esp

push ebx

push ecx

; Load the parameters into registers

mov eax, DWORD PTR [ebp+8] ; x

mov ebx, DWORD PTR [ebp+12] ; y

mov ecx, DWORD PTR [ebp+16] ; z

; Compute the product of x, y, and z

mul ebx

mul ecx

; Epilogue

pop ecx

pop ebx

mov esp, ebp

pop ebp

ret 12 ; Clean up the stack and return

ThreeProd ENDP

main PROC

push z

push y

push x

call ThreeProd

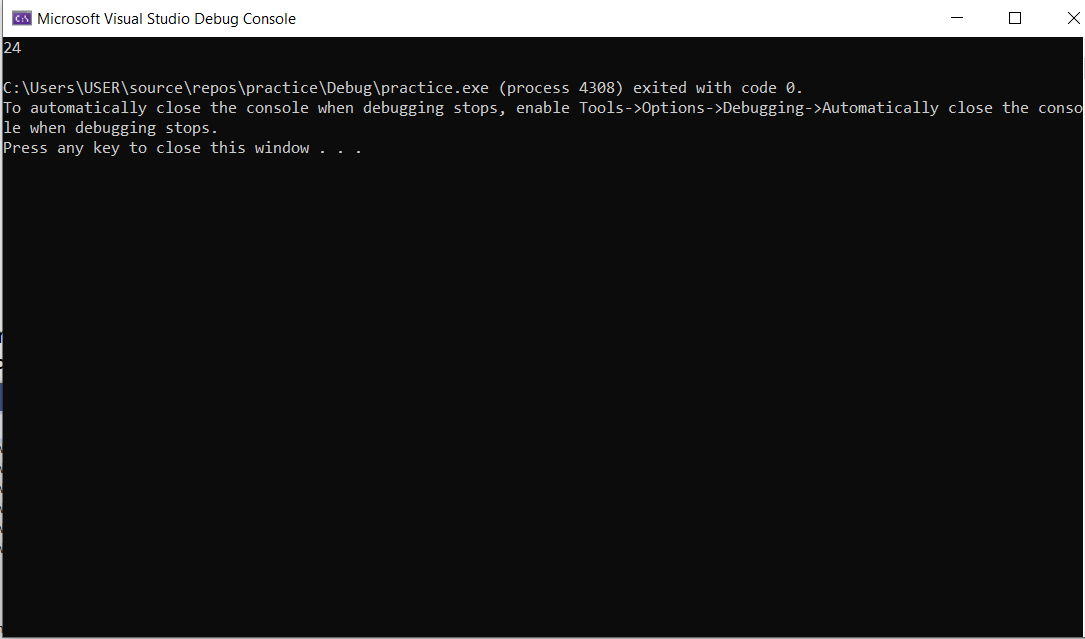
call WriteDec

call Crlf

exit

main ENDP

END main



## **Question 2:**

INCLUDE Irvine32.inc

.data

prompt1 BYTE "Enter the first number: ", 0

prompt2 BYTE "Enter the second number: ", 0

resultPrompt BYTE "The GCD is: ", 0

x DWORD ?

y DWORD ?

.code

TakeInput PROC

push ebp

mov ebp, esp

sub esp, 8

; Get the first number from the user

mov edx, OFFSET prompt1

call WriteString

mov eax, 0

call ReadInt

mov DWORD PTR [ebp-4], eax

; Get the second number from the user

mov edx, OFFSET prompt2

call WriteString

mov eax, 0

call ReadInt

mov DWORD PTR [ebp-8], eax

; Call the GCD procedure

push DWORD PTR [ebp-8] ; Push the second parameter onto the stack

push DWORD PTR [ebp-4] ; Push the first parameter onto the stack

call GCD

; Call the Display procedure

push eax ; Push the result onto the stack

call Display

mov esp, ebp

pop ebp

ret

TakeInput ENDP

GCD PROC

; Prologue

push ebp

mov ebp, esp

sub esp, 8

; Load the parameters into registers

mov eax, DWORD PTR [ebp+8] ; x

mov ebx, DWORD PTR [ebp+12] ; y

; Compute the GCD

cmp eax, ebx

jge gcd\_start

xchg eax, ebx ; Swap x and y

gcd\_start:

mov DWORD PTR [ebp-4], eax ; Store x in a local variable

mov DWORD PTR [ebp-8], ebx ; Store y in a local variable

mov edx, 0 ; Clear edx

gcd\_loop:

mov eax, DWORD PTR [ebp-4]

mov ebx, DWORD PTR [ebp-8]

xor edx, edx

div ebx

mov DWORD PTR [ebp-4], ebx

mov DWORD PTR [ebp-8], edx

cmp DWORD PTR [ebp-8], 0

jne gcd\_loop

mov eax, DWORD PTR [ebp-4] ; The GCD is now in eax

mov esp, ebp

pop ebp

ret 8

GCD ENDP

Display PROC

push ebp

mov ebp, esp

; Display the result

mov edx, OFFSET resultPrompt

call WriteString

mov eax, DWORD PTR [ebp+8]

call WriteDec

call Crlf

mov esp, ebp

pop ebp

ret

Display ENDP

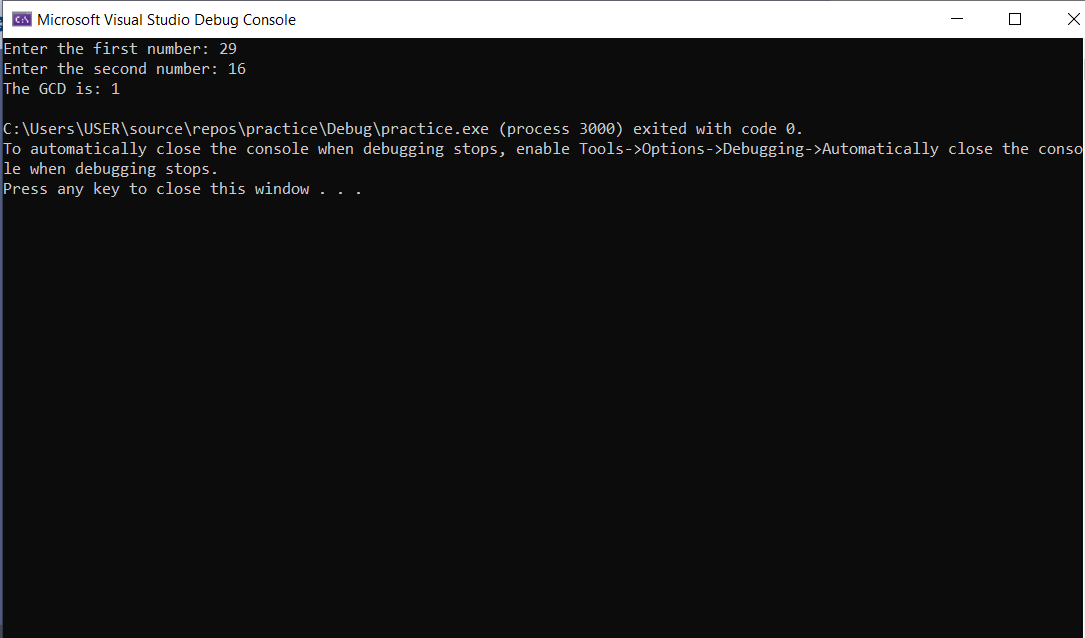
main PROC

call TakeInput

exit

main ENDP

END main



## **Question 3:**

Include Irvine32.inc

Include macros.inc

.data

Number DWORD ?

.code

MAIN PROC

mWrite "Enter the Number: "

call ReadInt

mov Number,eax

call LocalSquare

mWrite "The Square Of the Given number is: "

call WriteDec

exit

MAIN ENDP

LocalSquare PROC

Enter 4,0

mov eax,Number

mov edx,0

mov [ebp-4],eax

mov ebx,[ebp-4]

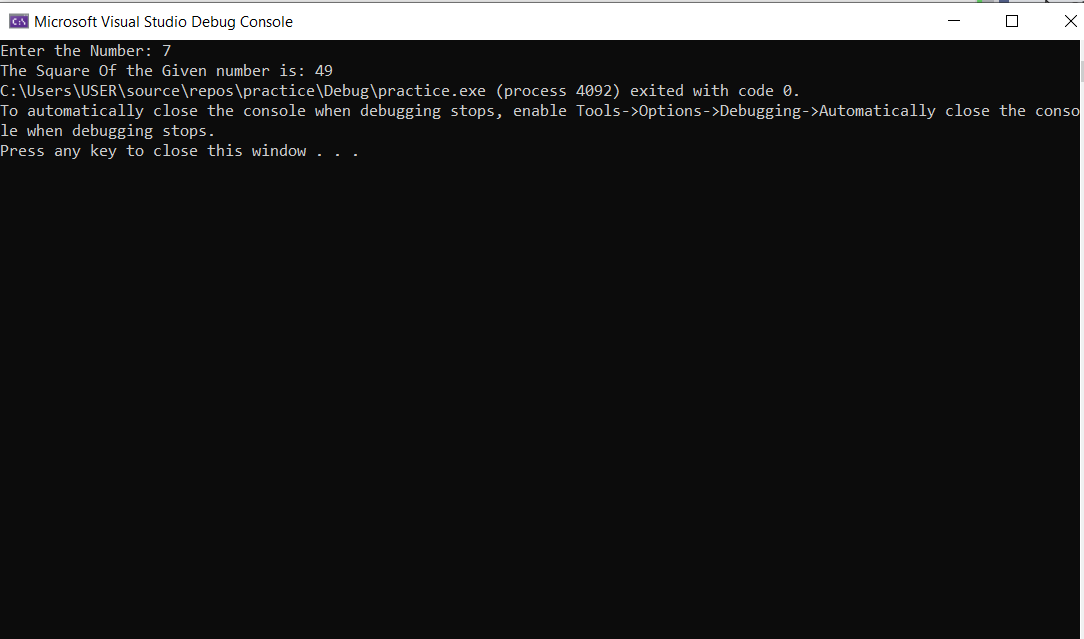
mul ebx

Leave

ret

LocalSquare ENDP

END MAIN



## **Question 4:**

Include Irvine32.inc

Include macros.inc

.data

Number DWORD ?

.code

MAIN PROC

mWrite "Enter the Number: "

call ReadInt

mov Number,eax

mov ebx,Number

dec ebx

call factorial

mWrite "Factorial Of the given number is: "

call WriteDec

exit

MAIN ENDP

factorial PROC

local multiplicand:DWORD

mov multiplicand,ebx

cmp ebx,0

mov edx,0

jz endRecursion

mul ebx

dec ebx

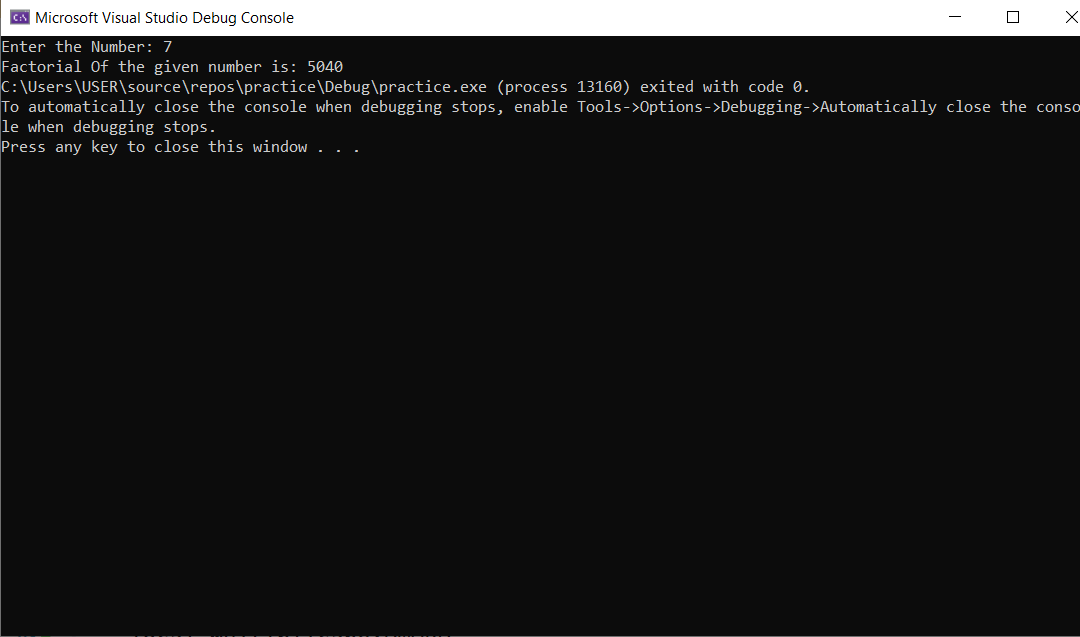
call factorial

endRecursion:

ret

factorial ENDP

END MAIN



## **Question 5:**

INCLUDE Irvine32.inc

INCLUDE macros.inc

.data

Number DWORD ?

StartTime DWORD ?

EndTime DWORD ?

RecursiveTime DWORD ?

IterativeTime DWORD ?

.code

MAIN PROC

mWrite "Enter the Number: "

call ReadInt

mov Number,eax

; Calculate factorial using recursion

mov ebx, Number

mov StartTime, eax

call factorialRecursive

mov EndTime, eax

mov RecursiveTime, edx

; Calculate factorial using iteration

mov eax, 1

mov ecx, Number

mov StartTime, eax

call factorialIterative

mov EndTime, eax

mov IterativeTime, edx

mWrite "Factorial using recursion: "

call WriteDec

call crlf

mWrite "Elapsed time: "

mov eax, RecursiveTime

call WriteDec

call crlf

mWrite " microseconds"

mWrite "Factorial using iteration: "

call WriteDec

call crlf

mov eax, IterativeTime

call WriteDec

mWrite " microseconds"

exit

MAIN ENDP

factorialRecursive PROC

cmp ebx, 1

jbe endRecursion

push ebx

dec ebx

call factorialRecursive

pop edx

mul edx

endRecursion:

ret

factorialRecursive ENDP

factorialIterative PROC

mov ecx, Number

mov eax, 1

factorialLoop:

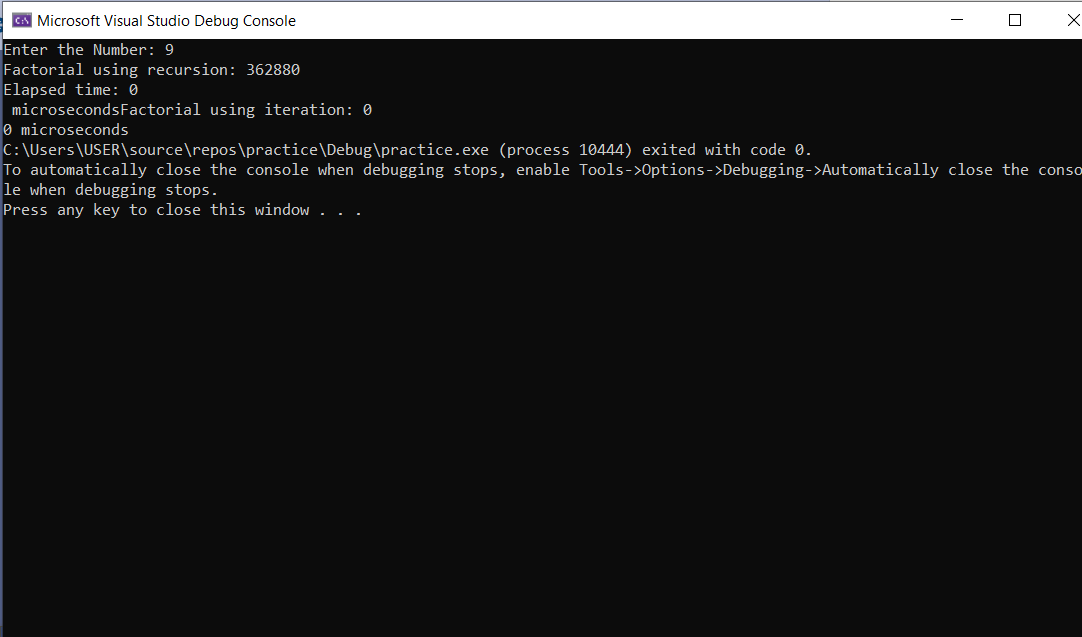
mul ecx

loop factorialLoop

ret

factorialIterative ENDP

END MAIN



## **Question 6:**

Include Irvine32.inc

Include macros.inc

.data

numList DWORD 4 DUP(?)

toCheckLarge DWORD 4 DUP(?) ;1 = all primes ;0 = any count is composite

PrimeCount DWORD 0

.code

MAIN PROC

mov ecx,4

mov ebx,0

getInput:

mWrite "Input value "

mov eax,ebx

inc eax

call WriteDec

mWrite " : "

call ReadInt

mov [numList + ebx\*TYPE numList],eax

push ebx

push ecx

call checkPrime

pop ecx

pop ebx

inc ebx

loop getInput

mov eax,primeCount

cmp eax,4

jz findLargest

jmp endProgram

findLargest:

call largestPrime

call crlf

mWrite "The Largest Prime among these numbers is: "

call WriteDec

call crlf

endProgram:

exit

MAIN ENDP

checkPrime PROC

local num:DWORD,limit:DWORD,current:DWORD

mov num,eax

cmp eax,2

jz isPrime

shr eax,1

mov limit,eax

mov ebx,2

mov current,ebx

mov ebx,0

mov ecx,limit

checkForPrime:

push ecx

mov edx,0

mov eax,num

mov ecx,current

div ecx

cmp edx,0

jz notAPrime

inc current

pop ecx

loop checkForPrime

jmp isPrime

notAPrime:

mWrite "It's not a prime number "

call crlf

jmp endFunction

isPrime:

mWrite "It's a prime number "

call crlf

inc PrimeCount

endFunction:

ret

checkPrime ENDP

largestPrime PROC

local largest:DWORD

mov largest,0

mov ebx,0

mov ecx,4

findLargest:

mov eax,[numList + ebx\* TYPE numList]

cmp eax,largest

jg newLargest

jmp continueLoop

newLargest:

mov largest,eax

continueLoop:

inc ebx

loop findLargest

ret

largestPrime ENDP

END MAIN

