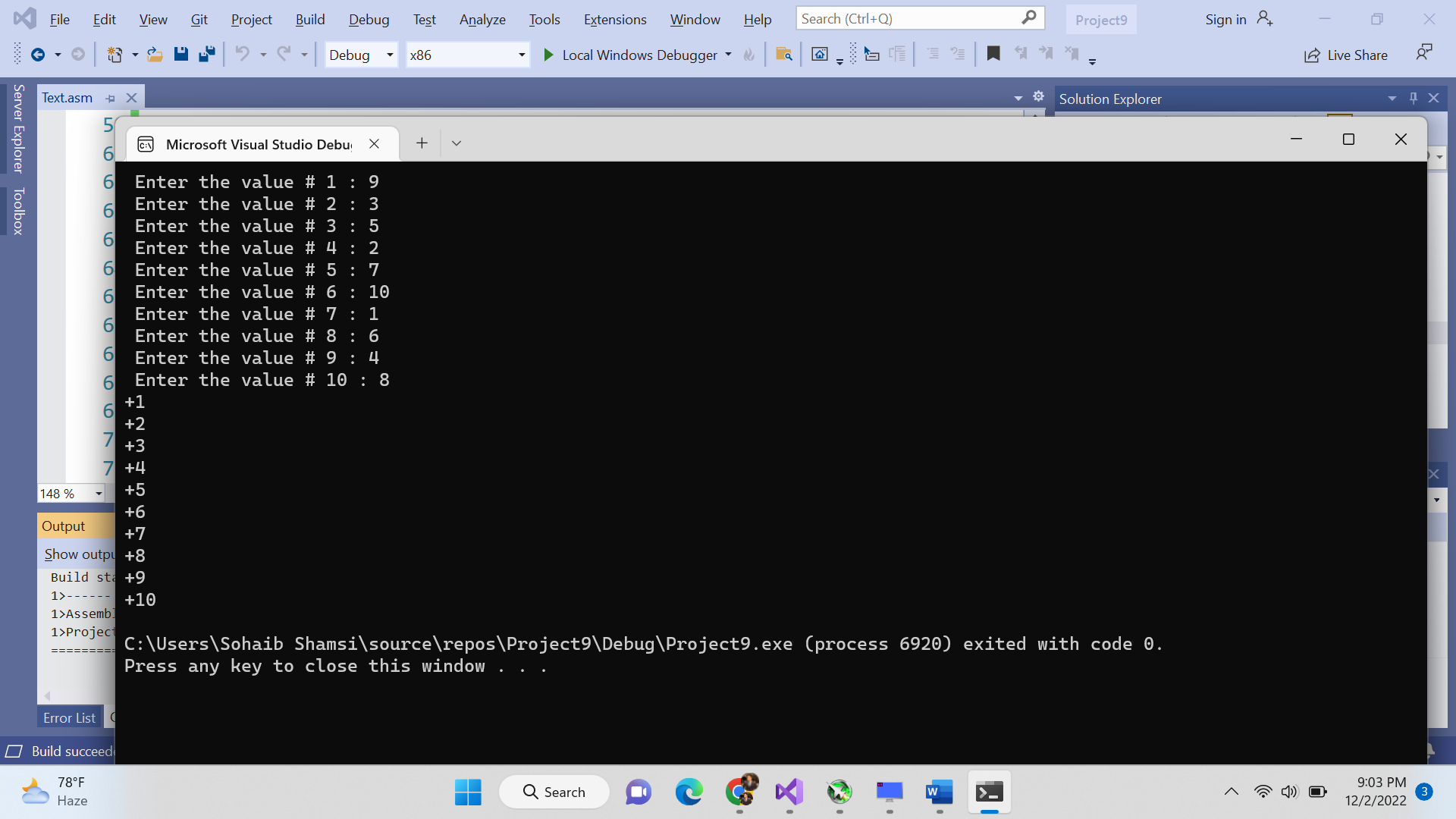
**LAB 10**

**Q1.**



Include Irvine32.inc

.data

arr SDWORD 10 DUP(?)

count DWORD 10

prompt1 byte " Enter the value # ", 0

colon byte " : ", 0

.code

MAIN PROC

mov ecx,10

mov ebx,0

inputValues:

mov edx, offset prompt1

call writestring

mov eax,ebx

inc eax

call WriteDec

mov edx, offset colon

call writestring

call ReadInt

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

inc ebx

loop inputValues

push OFFSET arr

push LENGTHOF arr

call BubbleSort

mov ebx,0

mov ecx,LENGTHOF arr

printArray:

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

call WriteInt

inc ebx

call crlf

loop printArray

exit

MAIN ENDP

BubbleSort PROC

push ebp

mov ebp,esp

mov esi,[ebp+12]

mov edi,[ebp+12]

mov ecx,[ebp+8]

dec ecx

mov ebx,0

outerLoop:

push ecx

mov esi,edi

innerLoop:

mov eax,[esi]

cmp eax,[esi+TYPE arr]

jg swapElements

continueLoop:

add esi,TYPE arr

loop innerLoop

pop ecx

loop outerLoop

jmp endProgram

swapElements:

mov eax,[esi]

xchg eax,[esi+TYPE arr]

mov [esi],eax

jmp continueLoop

endProgram:

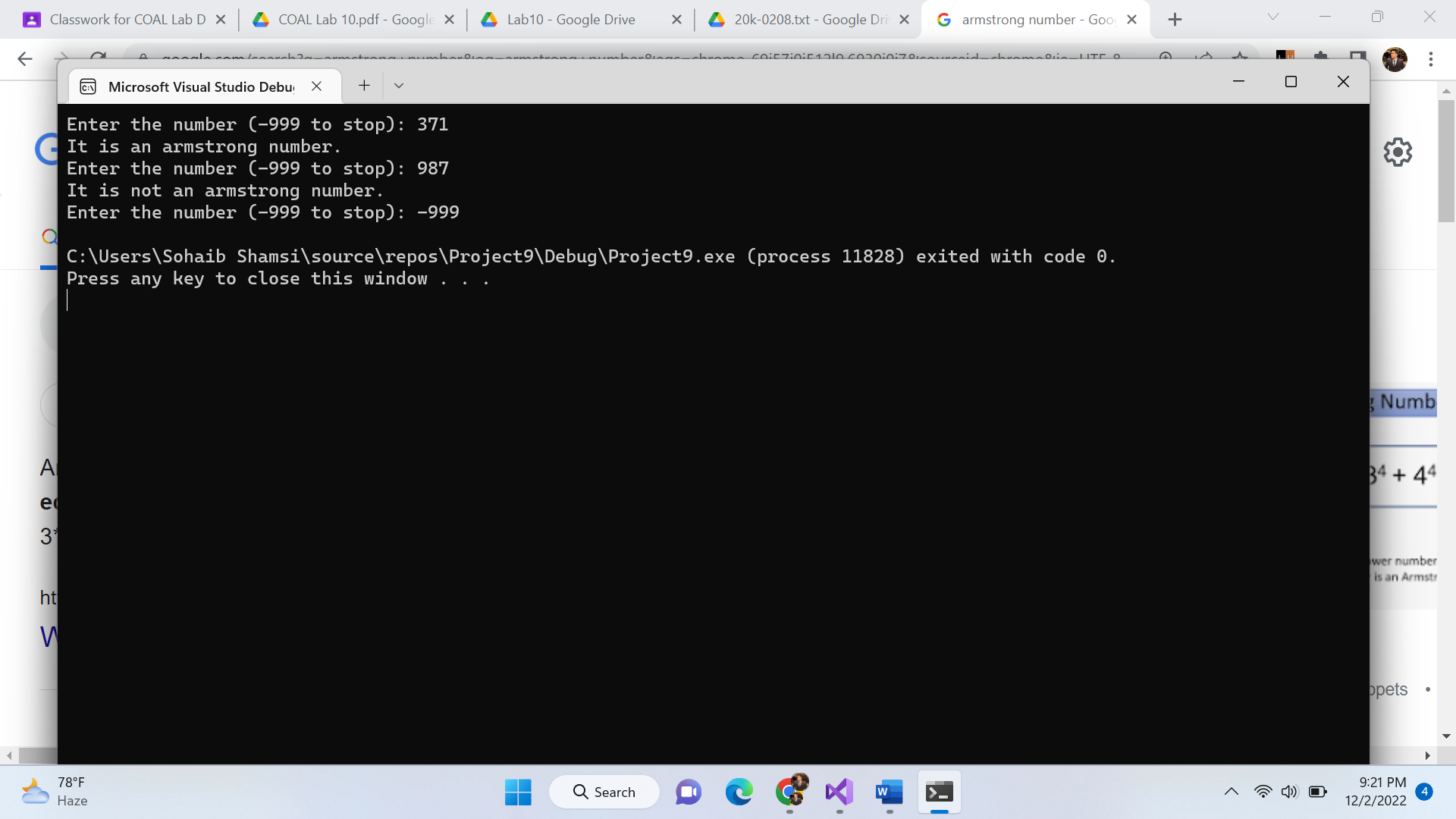
pop ebp

ret 8

BubbleSort ENDP

END MAIN

**Q2.**



TITLE Question#2

Include Irvine32.inc

.data

Number DWORD ?

prompt1 byte "Enter the number (-999 to stop): ", 0

out1 byte "It is an armstrong number. ", 0

out2 byte "It is not an armstrong number. ", 0

.code

MAIN PROC

infinitelyCallAndCheck:

mov ecx,0

mov edx, offset prompt1

call writestring

call ReadInt

cmp eax,-999

jz stopProgram

mov Number,eax

call checkArmstrong

loop infinitelyCallAndCheck

stopProgram:

exit

MAIN ENDP

checkArmstrong PROC

local Sum:DWORD,temp:DWORD,multiplicand:DWORD

mov Sum,0

mov ecx,0

mov eax,Number

mov ebx,eax

divideTillZero:

mov ecx,10

mov edx,0

div ecx

mov ebx,eax

mov temp,ebx

mov ebx,edx

mov ecx,2

mov eax,edx

mov multiplicand,edx

findCube:

mul multiplicand

loop findCube

add Sum,eax

mov eax,temp

cmp eax,0

jz endLoop

loop divideTillZero

endLoop:

mov eax,Sum

cmp eax,Number

jz isArmstrong

jmp isNotArmstrong

isArmstrong:

mov edx, offset out1

call writestring

call crlf

jmp endProgram

isNotArmstrong:

mov edx, offset out2

call writestring

call crlf

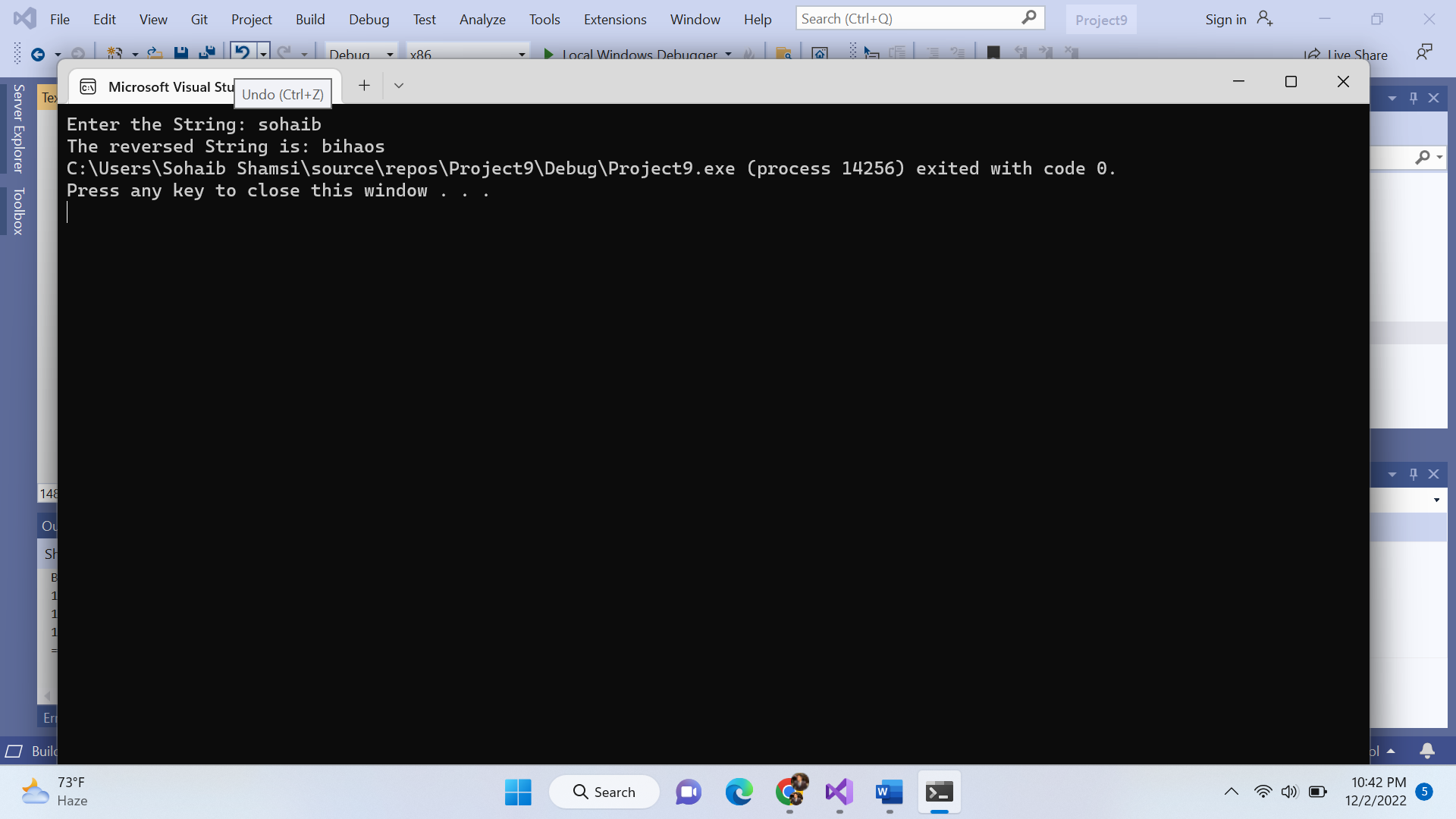
endProgram:

ret

checkArmstrong ENDP

END MAIN

**Q3.**



Include Irvine32.inc

.data

string BYTE 100 DUP(?)

sizeArray DWORD ?

prompt1 byte "Enter the String: ", 0

prompt2 byte "The reversed String is: ", 0

.code

MAIN PROC

mov edx, offset prompt1

call writestring

mov edx,OFFSET string

mov ecx,100

call ReadString

mov sizeArray,eax

mov [string+eax],0

mov ebx,0

mov edx,sizeArray

dec edx

mov ecx,0

call Reverse

mov edx, offset prompt2

call writestring

mov edx,OFFSET string

call WriteString

exit

MAIN ENDP

Reverse PROC

inc ecx

mov eax,sizeArray

shr eax,1

inc eax

cmp ecx,eax

jz WorkTime

call Reverse

WorkTime:

mov al,[string + ebx\*TYPE string]

xchg al,[string + edx\*TYPE string]

mov [string+ebx\*TYPE string],al

dec edx

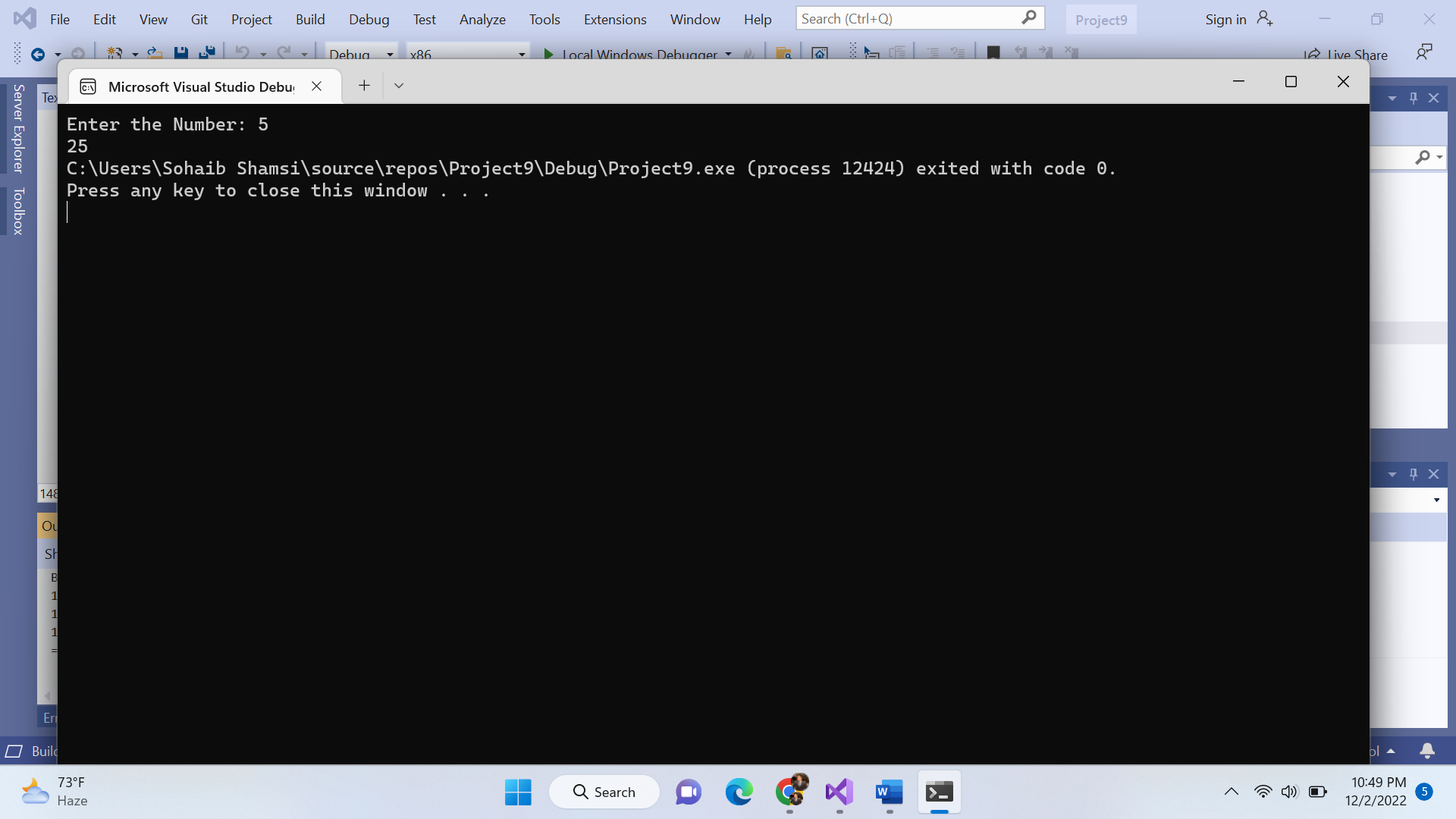
inc ebx

ret

Reverse ENDP

END MAIN

**Q4.**



Include Irvine32.inc

.data

Number DWORD ?

prompt1 byte "Enter the Number: ", 0

.code

MAIN PROC

mov edx, offset prompt1

call writestring

call ReadInt

mov Number,eax

call LocalSquare

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

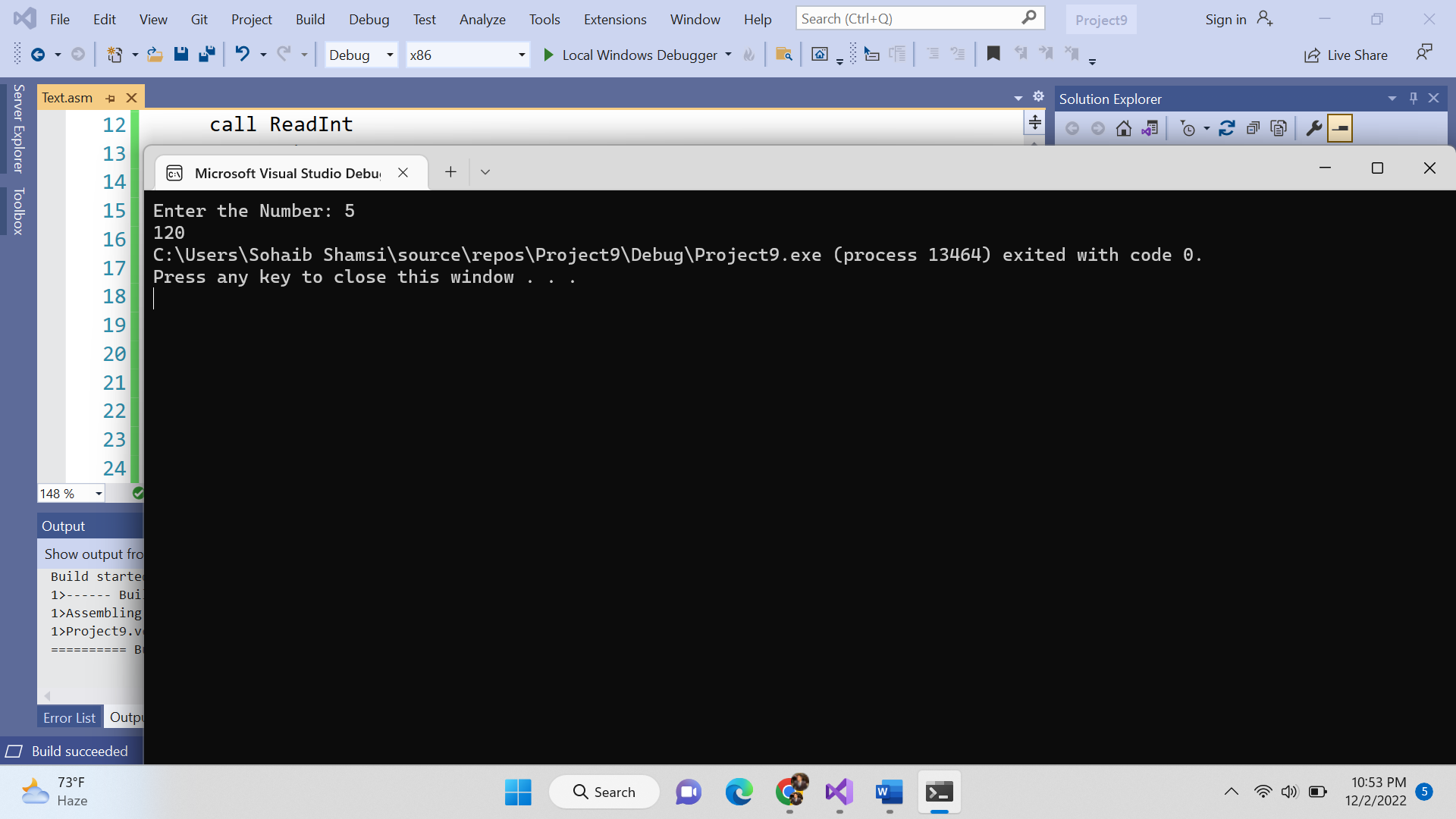
Leave

ret

LocalSquare ENDP

END MAIN

**Q5.**



TITLE QUESTION # 4

Include Irvine32.inc

.data

Number DWORD ?

prompt1 byte "Enter the Number: ", 0

.code

MAIN PROC

mov edx, offset prompt1

call writestring

call ReadInt

mov Number,eax

mov ebx,Number

dec ebx

call factorial

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

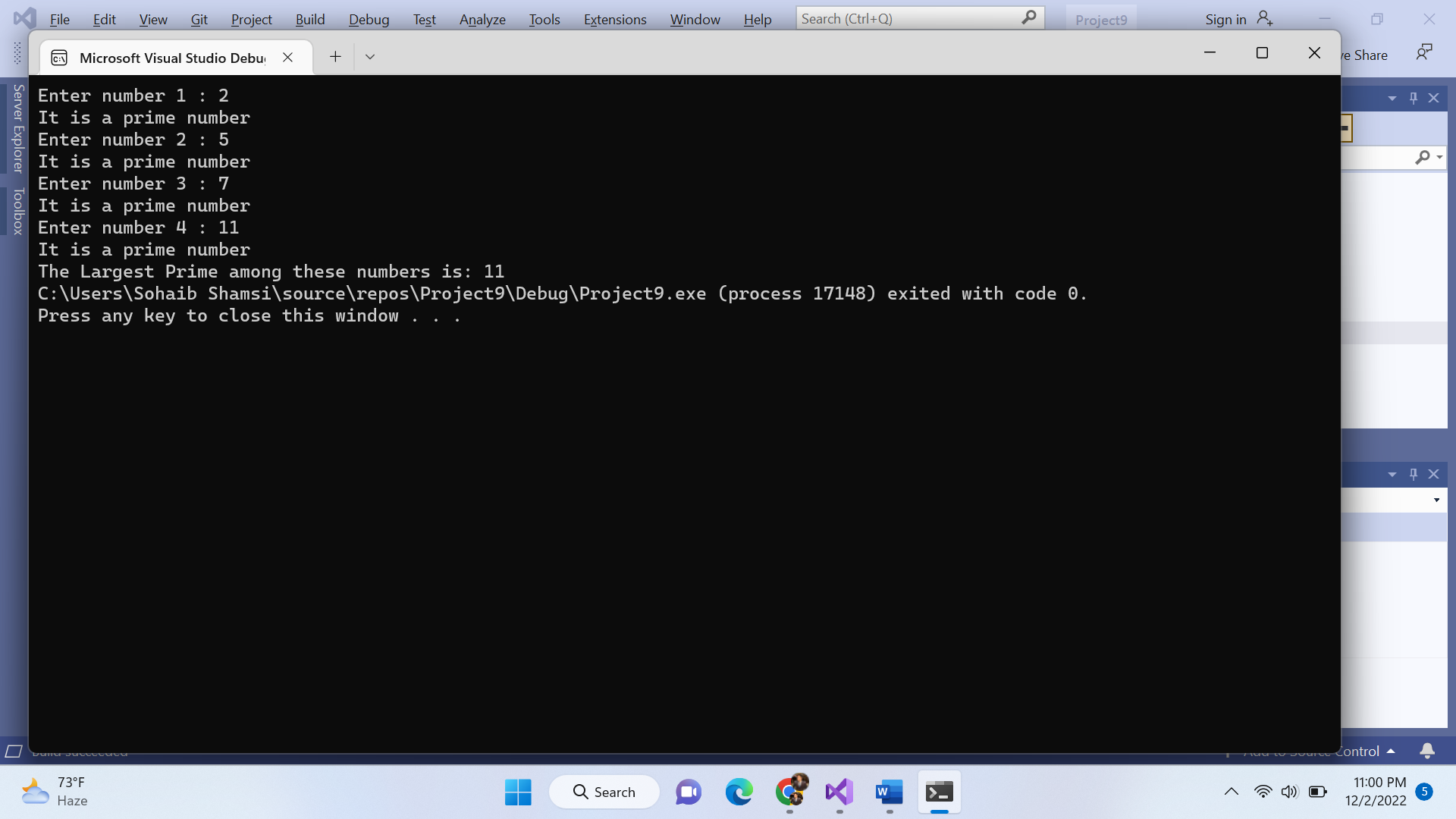
endRecursion:

ret

factorial ENDP

END MAIN

**Q6.**



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 "Enter number "

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

mWrite "The Largest Prime among these numbers is: "

call WriteDec

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 is not a prime number "

call crlf

jmp endFunction

isPrime:

mWrite "It is 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