**LAB 12**

**21K-3278**

**Sohaib Shamsi**

**Q1.**

.686 ;Target processor. Use instructions for Pentium class machines

.MODEL FLAT, C ;Use the flat memory model. Use C calling conventions

.STACK 2048 ;Define a stack segment of 1KB (Not required for this example)

.DATA ;Create a near data segment. Local variables are declared after

;this directive (Not required for this example)

.CODE ;Indicates the start of a code segment.

var\_1 dword 10

str\_1 byte 50,100,34,5,6,78,12,45,67

str\_2 byte 5000 dup(?)

.CODE ;Indicates the start of a code segment.

clear PROC

xor eax, eax

xor ebx, ebx

ret

clear ENDP

END

#include <stdio.h>

#include <iostream>

// extern "C" instruct the compiler to use C calling conventions

extern "C" void clear();

int threeProd(int x, int y, int z)

{

int f\_ans;

\_asm

{

mov eax, x

imul y

imul z

mov f\_ans, eax

}

return f\_ans;

}

int main()

{

clear();

//define variables

int a, b, c;

printf("Enter three numbers: \n");

std ::cin >> a >> b >> c;

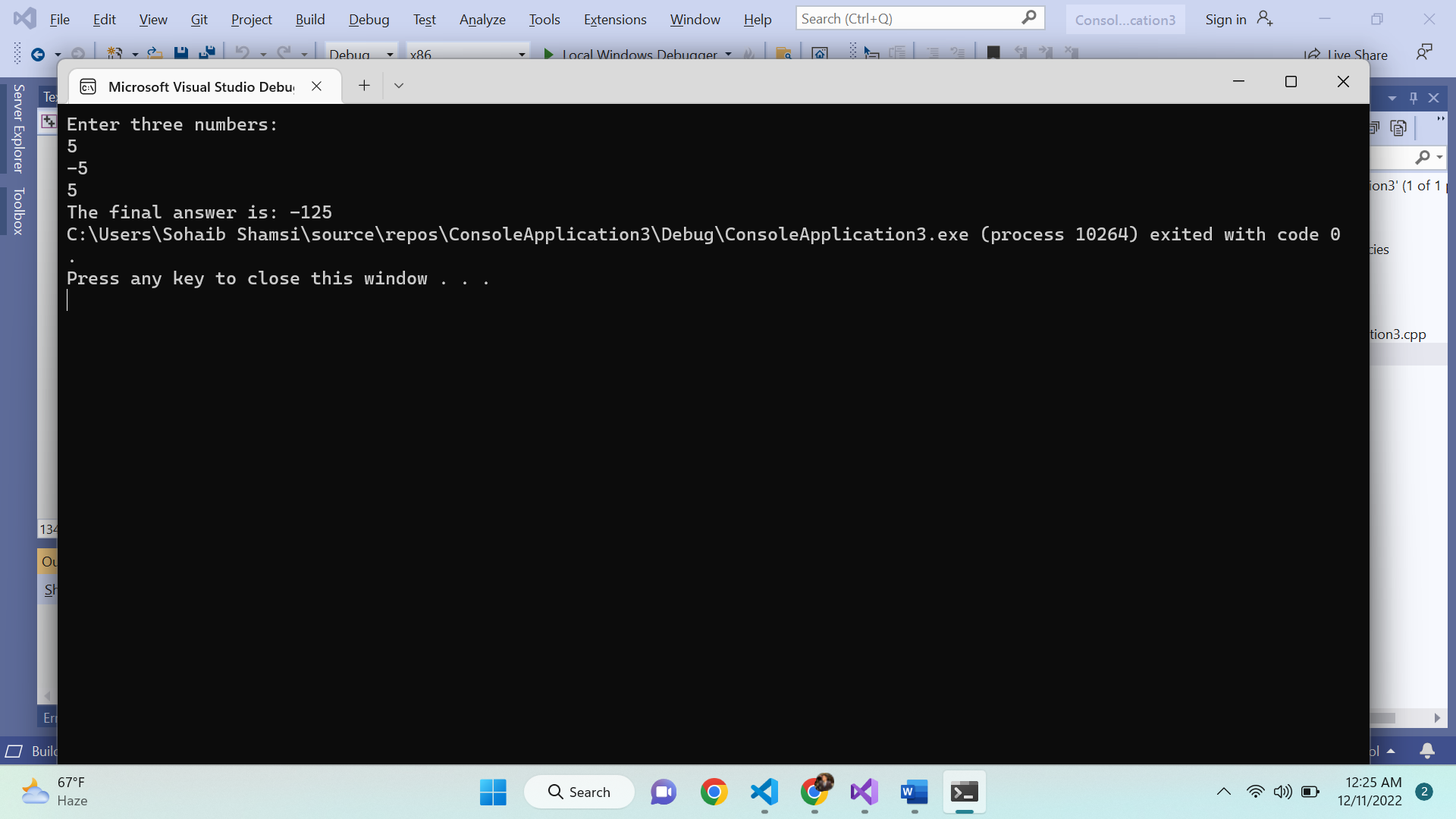
//switch to assembly

int ans = threeProd(a, b, c);

std::cout << "The final answer is: " << ans;

return 0;

}



**Q2.**

.686 ;Target processor. Use instructions for Pentium class machines

.MODEL FLAT, C ;Use the flat memory model. Use C calling conventions

.STACK 2048 ;Define a stack segment of 1KB (Not required for this example)

.DATA ;Create a near data segment. Local variables are declared after

;this directive (Not required for this example)

.CODE ;Indicates the start of a code segment.

GCD PROC

LOCAL A:DWORD,B:DWORD

mov eax,[ebp+8] ; arr[0]

mov A,eax

mov eax,[ebp+12] ; arr[1]

mov B,eax

cmp A,0

jz firstEnd

cmp B,0

jz secondEnd

mov eax,A

cmp eax,B

jz thirdEnd

jg firstRecurse

jl secondRecurse

firstEnd:

mov eax,B

pop ebp

ret

jmp endProgram

secondEnd:

mov eax,A

pop ebp

ret

jmp endProgram

thirdEnd:

mov eax,A

ret

jmp endProgram

firstRecurse:

mov eax,A

sub eax,B

mov A,eax

push B

push A

call GCD

jmp endProgram

secondRecurse:

mov eax,B

sub eax,A

mov B,eax

push B

push A

call GCD

endProgram:

ret

GCD ENDP

END

END

#include <stdio.h>

#include <iostream>

extern "C" void GCD(int, int);

int main()

{

int arr[2];

std ::cout << "Enter the two numbers: ";

std ::cin >> arr[0];

std::cin >> arr[1];

GCD(arr[0], arr[1]);

int gcd;

\_asm {

mov gcd, eax

}

std :: cout << "GCD: " << gcd;

return 0;

}

