**COAL LAB 10**

**Question 1**

Include Irvine32.inc

.data

arr1 SDWORD 5, -9, 12, 1, -7, 8

.code

Main PROC

push offset arr1

push lengthof arr1

call BubbleSort

MOV ecx, lengthof arr1

MOV esi, offset arr1

PRINT:

MOV eax, [esi]

call Writeint

Call crlf

ADD esi, 4

loop print

exit

main endp

BubbleSort PROC

push ebp

MOV ebp, esp

MOV ecx, [ebp+8]

MOV esi, [ebp+12]

L1:

MOV edi, esi

MOV eax, [esi]

MOV ebx, ecx

L2:

cmp [edi], eax

JGE around

MOV edx, eax

MOV eax, [edi]

MOV [edi], edx

MOV [esi], eax

around:

ADD edi, 4

loop L2

ADD esi, 4

MOV ecx, ebx

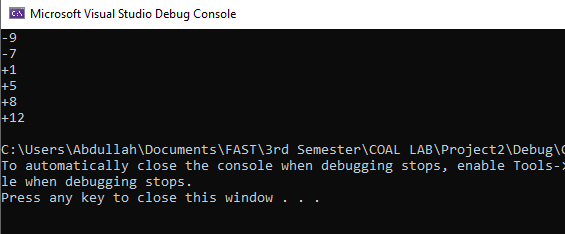
Loop L1

pop ebp

ret 8

BubbleSort endp

end main

****

**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

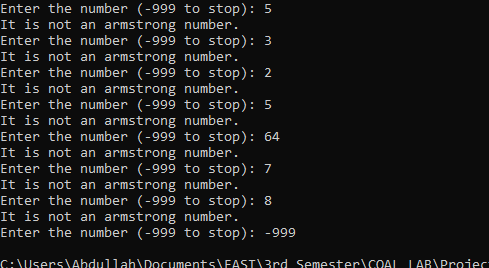
call crlf

endProgram:

ret

checkArmstrong ENDP

END MAIN

****

**Question 3**

Include Irvine32.inc

.data

str1 BYTE "It's a good life we lead.",0

.code

Main PROC

PUSH OFFSET str1

push lengthof str1

xor eax, eax

xor ebx, ebx

call Reverse

exit

main endp

Reverse PROC

push ebp

MOV ebp, esp

MOV esi, [ebp+12]

MOV ecx, [ebp+8]

MOV al, [str1+ecx]

call writechar

cmp ecx, 0

JL endd

dec ecx

push ecx

call Reverse

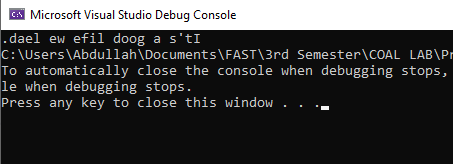
endd:

pop ebp

ret 4

Reverse endp

end main

****

**Question 4**

Include Irvine32.inc

.data

str1 BYTE "Enter a number: ",0

.code

Main PROC

PUSH OFFSet str1

call

exit

main endp

LocalSqaure PROC

enter 4,0

xor eax, eax

MOV edx, [ebp+8]

call writestring

call readdec

MOV [ebp-4], eax

MOV ebx, eax

imul bl

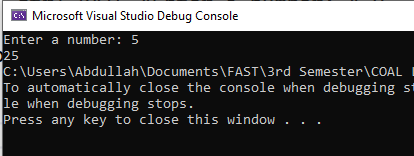
call writedec

leave

ret

LocalSqaure endp

end main

****

**Question 5**

Include Irvine32.inc

.code

Main PROC

push 5

call Fact

call writedec

exit

main endp

Fact PROC

enter 0,0

MOV eax, [ebp+8]

cmp eax, 1

JE endd

dec eax

push eax

call Fact

endd:

MOV ebx, [ebp+8]

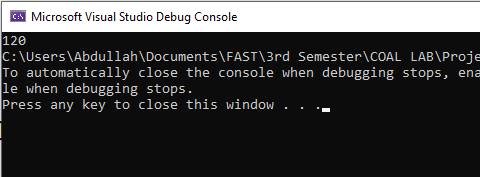
imul bl

leave

ret 4

Fact endp

end main

****

**Question 6**

Include Irvine32.inc

.data

str1 BYTE "Enter 4 numbers: ",0

str2 BYTE "All numbers are not prime.",0

str3 BYTE "The largest prime number is: ",0

nums DWORD 4 dup(?)

Prime\_count DWORD 0

.code

Main PROC

xor eax, eax

MOV ecx, 4

MOV edx, offset str1

call writestring

MOV esi, offset nums

L1:

call readdec

MOV [esi], eax

ADD esi, 4

loop L1

push offset nums

call CheckPrime

MOV eax, 4

cmp Prime\_count, eax

JNE final

push offset nums

call LargestPrime

exit

final:

mov edx, offset str2

call writestring

exit

main endp

CheckPrime PROC

enter 0,0

MOV esi, [ebp+8]

MOV edx, 0

MOV ecx, 4

ajeeb:

push ecx

MOV edx, 0

MOV ecx, [esi]

dec ecx

L1:

MOV eax, [esi]

MOV ebx, ecx

idiv bl

cmp ah, 0

JNE endd

inc edx

endd:

loop l1

cmp edx, 1

JNE cond2

inc prime\_count

cond2:

pop ecx

ADD esi, 4

loop ajeeb

leave

ret 4

CheckPrime endp

LargestPrime PROC

enter 0,0

MOV eax, -9999

MOV esi, [ebp+8]

MOV ecx, 4

gret:

cmp [esi], eax

JLE cond3

MOV eax, [esi]

cond3:

ADD esi,4

loop gret

MOV edx, offset str3

call writestring

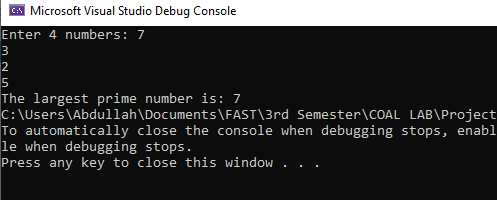
call writedec

leave

ret 4

LargestPrime endp

end main

****