**COAL LAB 10**

**Q1**

Include Irvine32.inc

.data

array dword 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

.code

main PROC

call BubbleSort

exit

main endp

BubbleSort PROC

mov eax,0

mov ebx,0

mov ecx,0

mov edx,0

mov esi,0

mov edi,0

mov ecx,11

L1:

mov edx,ecx

mov ecx,10

L2:

mov eax,0

mov ebx,0

mov eax,array[edi\*type array]

inc edi

mov ebx,array[edi\*type array]

dec edi

cmp eax,ebx

JG call\_swap

back\_after\_swap:

inc edi

loop L2

mov edi,0

inc esi

mov ecx,edx

loop L1

call print\_array

ret

call\_swap:

call swap

jmp back\_after\_swap

BubbleSort endp

swap PROC

xchg eax,ebx

mov array[edi\*type array],eax

inc edi

mov array[edi\*type array],ebx

dec edi

mov eax,0

mov ebx,0

ret

swap endp

print\_array PROC

mov eax,0

mov esi,0

mov ecx,11

mov edx,0

L3:

mov eax,array[edx\*type array]

call writedec

call crlf

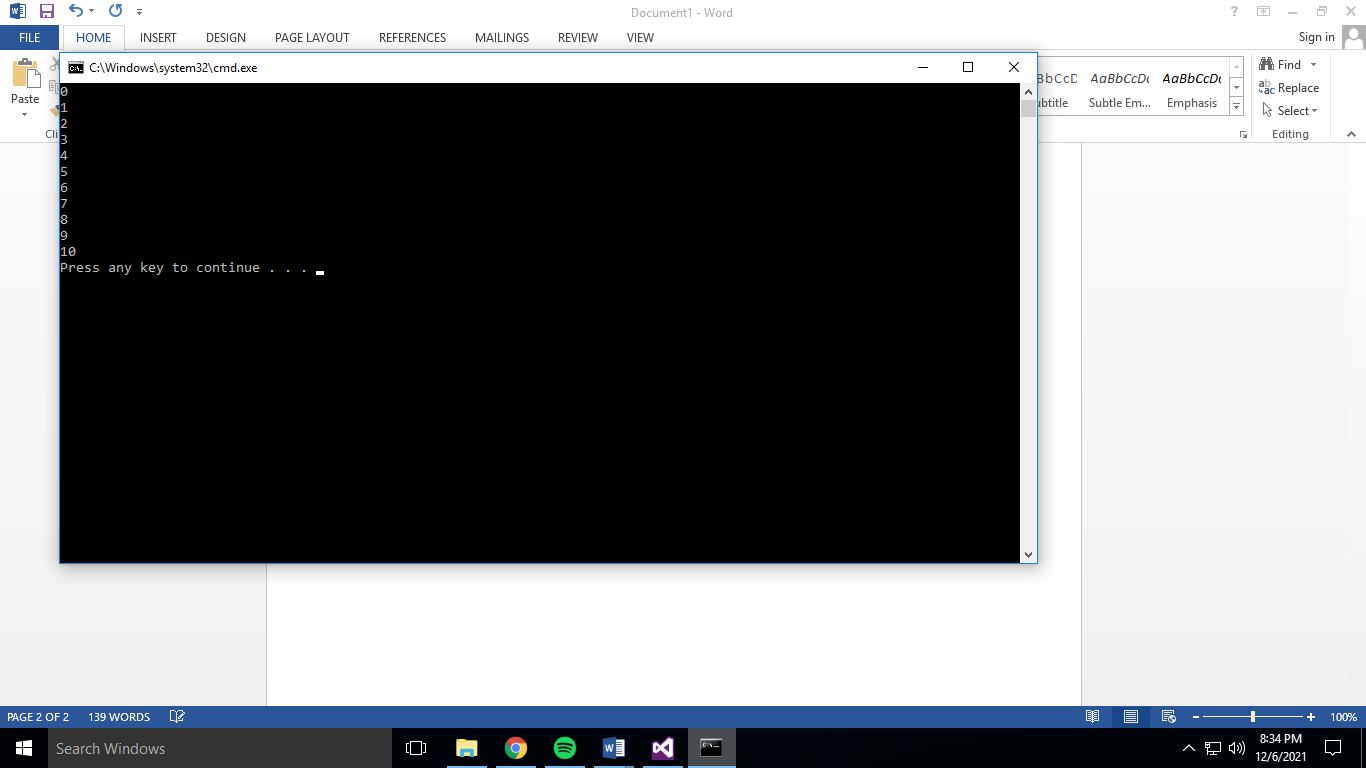
inc edx

loop L3

ret

print\_array endp

end main



**Q2**

Include Irvine32.inc

.data

input\_string byte 'Enter 3 digit Integer :',0

armstrong\_string byte 'The entered number is Armstrong',0

armstrong\_not\_string byte 'The entered number is not Armstrong',0

.code

main PROC

call TakeInput

exit

main endp

TakeInput PROC Uses edx

mov edx,offset input\_string

call writestring

mov eax,0

call readdec

call Armstrong

call display

ret

TakeInput ENDP

Armstrong PROC USES eax

LOCAL remainder:byte , quotient:byte , divisor:byte , numbers:byte , cubes : dword

mov esi,0

mov ecx,3

L1: ;using this loop to get each digit from the number

mov divisor,10d

div divisor

mov quotient,al

mov remainder[esi],ah

inc esi

mov eax,0

movzx ax,quotient

loop L1

mov eax,0

mov ecx,0

mov ecx,3

mov eax,0

mov esi,0

L2: ;using this loop to calculate cubes of the number

mov al,remainder[esi]

call calculate\_cube

mov cubes[esi \* type cubes ],eax

inc esi

mov eax,0

loop L2

mov ecx,0

mov ecx,3

mov esi,0

mov edx,0

L3: ;adding cubes of the numbers

add edx,cubes[esi \* type cubes]

inc esi

loop L3

ret

Armstrong ENDP

calculate\_cube PROC uses ecx

LOCAL multiplier:byte

mov ecx,0

mov ecx,2

mov multiplier,al

L1:

mul multiplier

loop L1

ret

calculate\_cube ENDP

display PROC

cmp eax,edx

JE display\_equal

mov edx,offset armstrong\_not\_string

call writestring

ret

display\_equal:

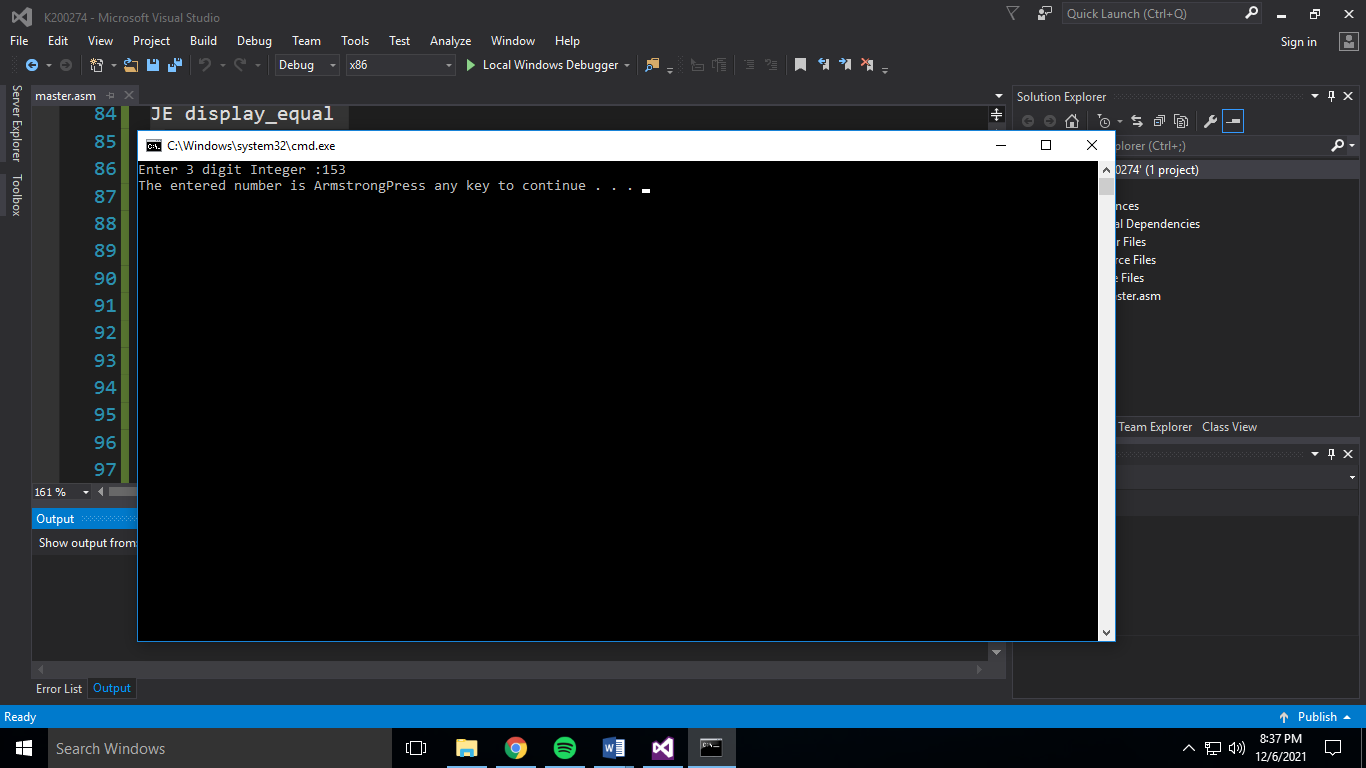
mov edx,offset armstrong\_string

call writestring

ret

display ENDP

end main



**Q3**

Include Irvine32.inc

.data

input\_string byte 'Nashit\_Budhwani',0

.code

main PROC

mov ecx,0

mov esi,0

mov ecx,lengthof input\_string

mov esi,lengthof input\_string

dec esi

call reverse

exit

main endp

reverse PROC

cmp ecx,0

JE return\_func

dec esi

mov eax,0

mov al,input\_string[esi]

call writechar

dec ecx

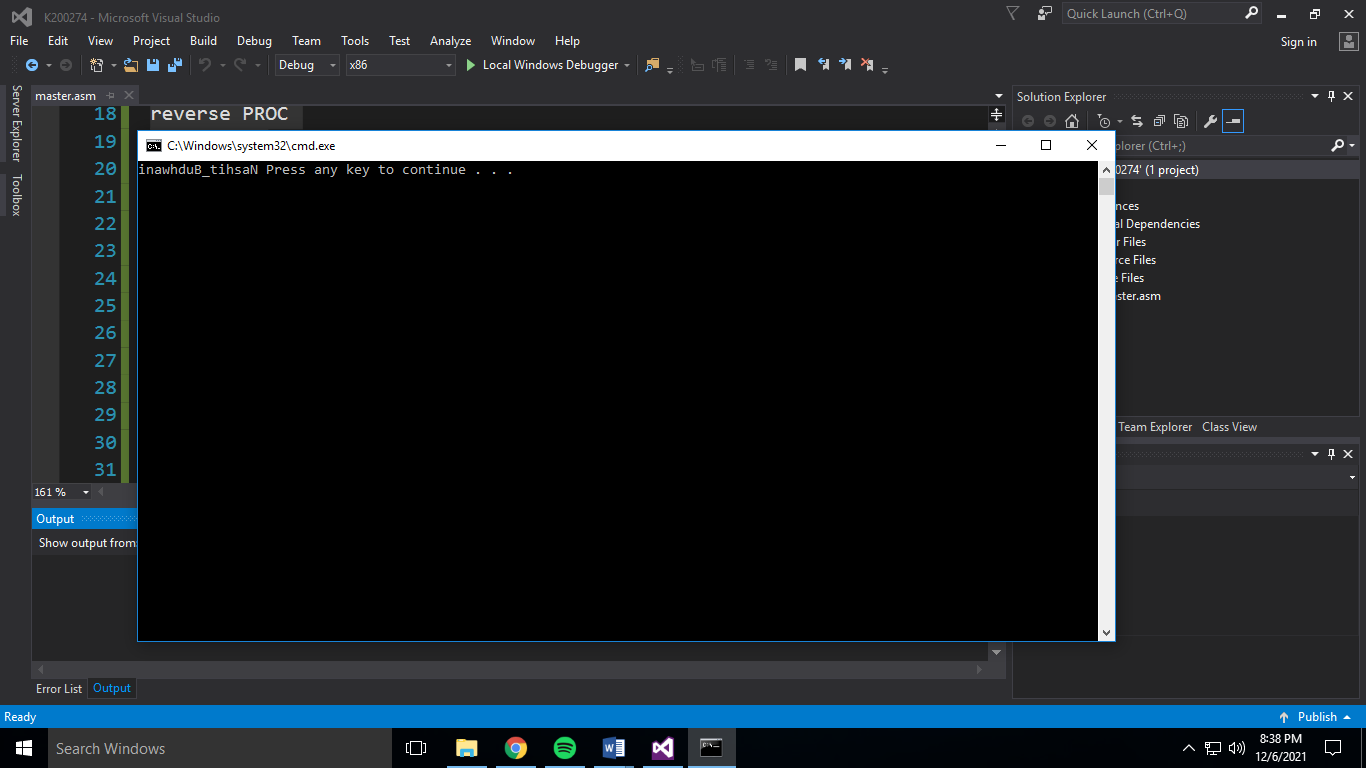
call reverse

return\_func:

ret

reverse ENDP

end main



**Q4**Include Irvine32.inc

.code

main PROC

mov eax,0

call readdec

push eax

call AddTwo

call writedec

exit

main endp

AddTwo PROC

enter 0, 0

mov eax, [ebp + 8]

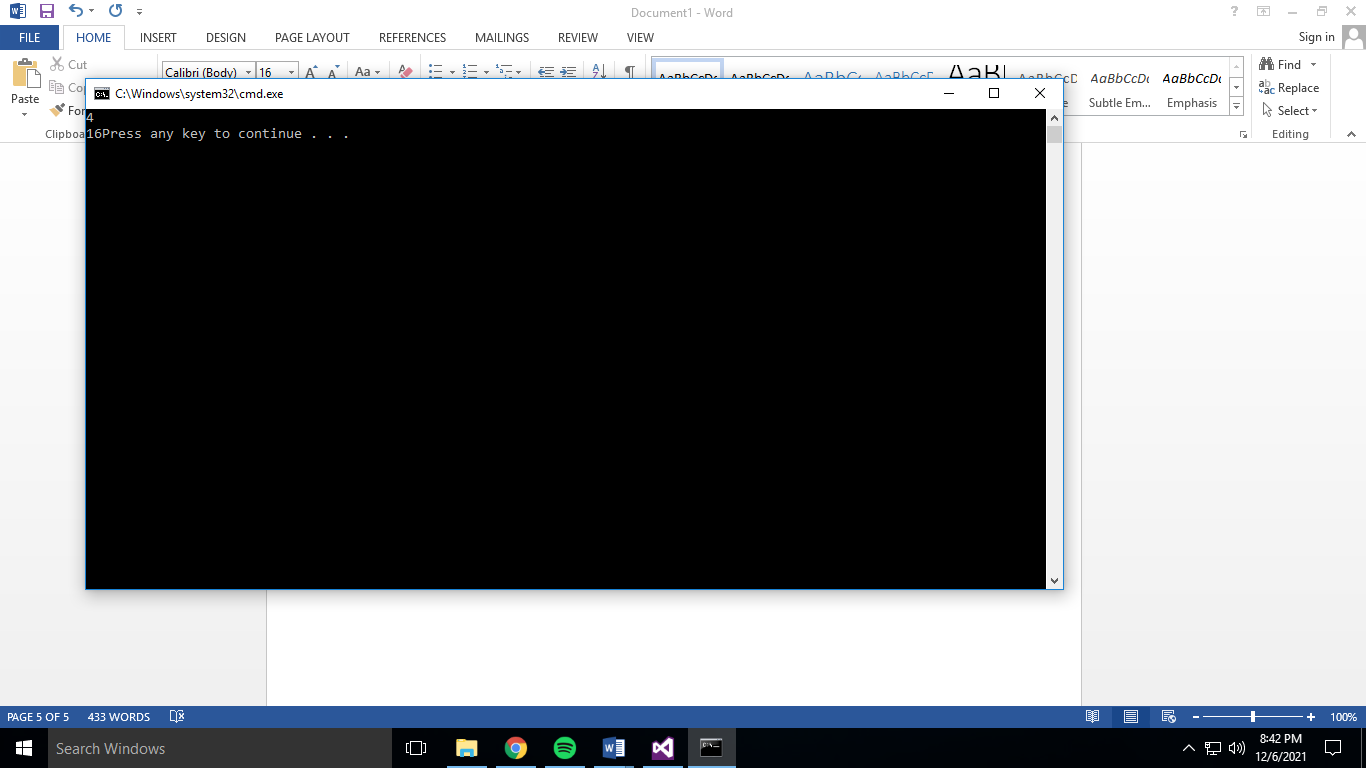
mul al

leave

ret

AddTwo ENDP

end main



**Q5**

Include Irvine32.inc

.code

main PROC

mov eax,0

mov ebx,0

mov edx,0

call readdec

push eax

call Factorial

call writedec

exit

main endp

Factorial PROC

enter 0,0

mov ebx,0

mov ebx,[ebp+8]

cmp ebx,1

JE ret\_func

JL ret\_one

dec ebx

mul bl

push ebx

call Factorial

ret\_func:

leave

ret 4

ret\_one:

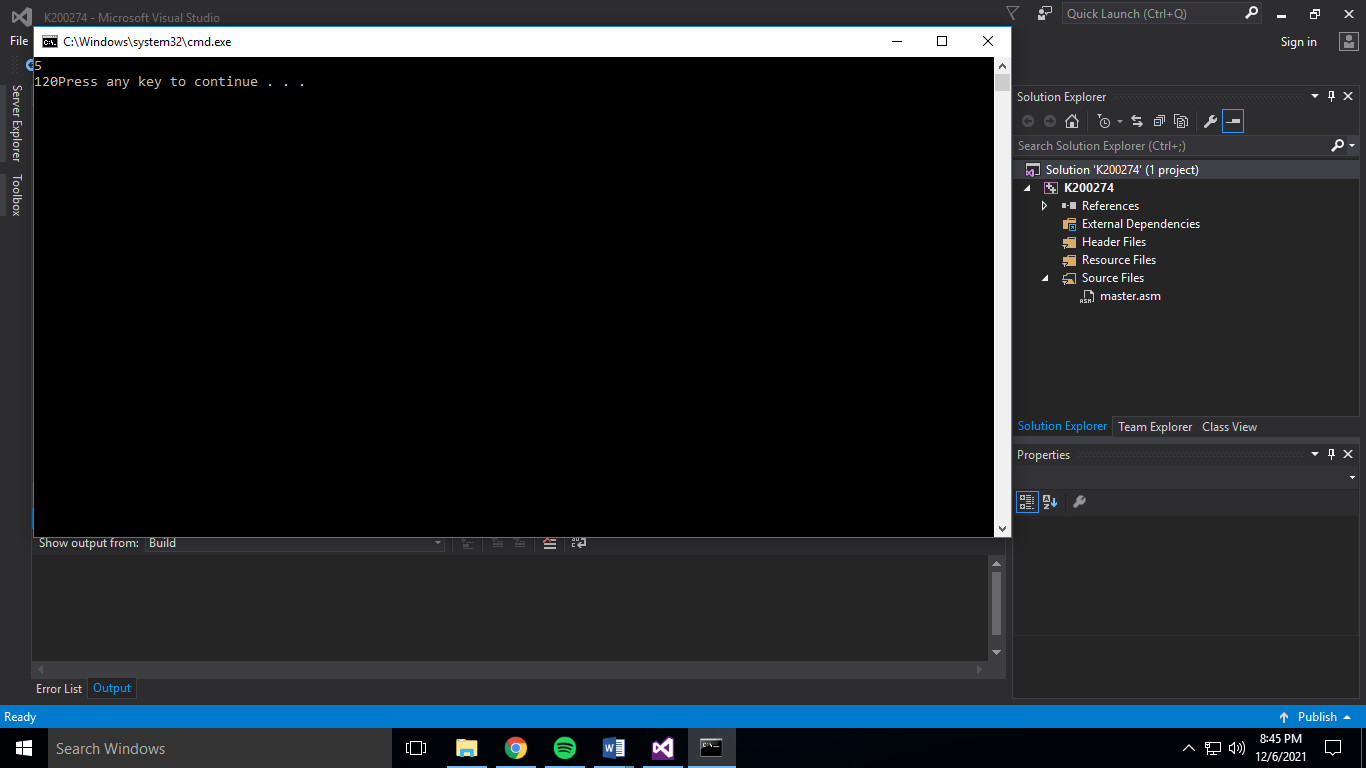
leave

mov eax,1

ret 4

Factorial ENDP

end main



**Q6**

Include Irvine32.inc

.data

not\_prime\_str byte 'Not A Prime Number',0

greatest\_prime\_str byte 'The largest prime number is : ',0

array dword 0,0,0

.code

main PROC

mov esi,0

mov ecx,4

L1:

mov eax,0

call readdec

mov array[esi\*type array],eax

inc esi

call checkPrime

loop L1

cmp eax,0

JE not\_prime

call greatest\_prime

exit

not\_prime:

mov edx,offset not\_prime\_str

call writestring

exit

main endp

greatest\_prime PROC

LOCAL max:dword

mov max,0

mov ecx,0

mov ecx,lengthof array

mov esi,0

L3:

mov eax,array[esi\*type array]

cmp eax,max

JG update\_max

back\_after:

inc esi

loop L3

mov edx,offset greatest\_prime\_str

call writestring

mov eax,max

call writedec

ret

update\_max:

mov max,eax

jmp back\_after

greatest\_prime ENDP

checkPrime PROC

LOCAL divisor:byte

mov divisor,2

div divisor

cmp ah,0

JE false\_l

mov eax,1

ret

false\_l:

mov eax,0

ret

checkPrime ENDP

end main

