**COAL LAB 10**

**21k-4834**

**TASK 1**

INCLUDE Irvine32.inc

INCLUDE macros.inc

.data

arr Dword 6,5,9,7,4

prompt1 BYTE "Array before sort: ",0

promtp2 BYTE "Array after sort: ",0

temp dword ?

sizea Dword 5

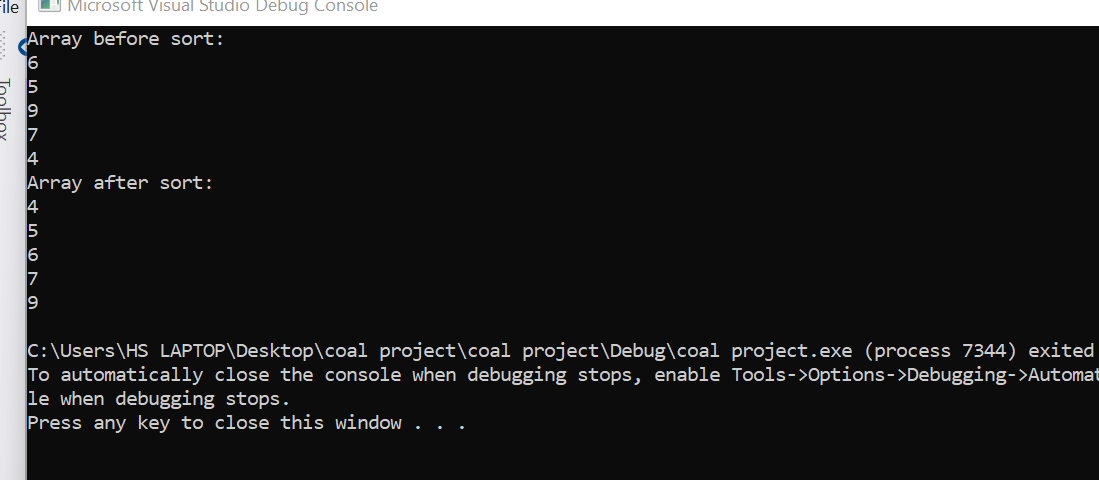
.code

main proc

mov edx,offset prompt1

call writestring

call crlf



mov ecx,5

mov esi,offset arr

l1:

mov eax,[esi]

call writedec

call crlf

add esi,type arr

loop l1

push offset arr

push sizea

call bubble

mov edx,offset promtp2

call writestring

call crlf

mov ecx,5

mov esi,offset arr

l4:

mov eax,[esi]

call writedec

call crlf

add esi,type arr

loop l4

exit

main endp

bubble proc

push ebp

mov ebp,esp

mov esi,[ebp+12]

mov ecx,[ebp+8]

l1:

push ecx

mov ecx,4

l2:

mov eax,[esi]

cmp eax,[esi + 4]

ja swap

add esi,type arr

loop l2

mov esi,offset arr

pop ecx

loop l1

jmp l3

swap:

xchg eax,[esi+4]

mov [esi],eax

jmp l2

l3:

pop ebp

ret

bubble endp

end main

**TASK 2**

include irvine32.inc

.data

var1 dword ?

var2 dword ?

var3 dword ?

valid byte "Armstrong-True",0

invalid byte "Armstrong-False",0

enters byte "Enter 3 digit one by one: ",0

.code

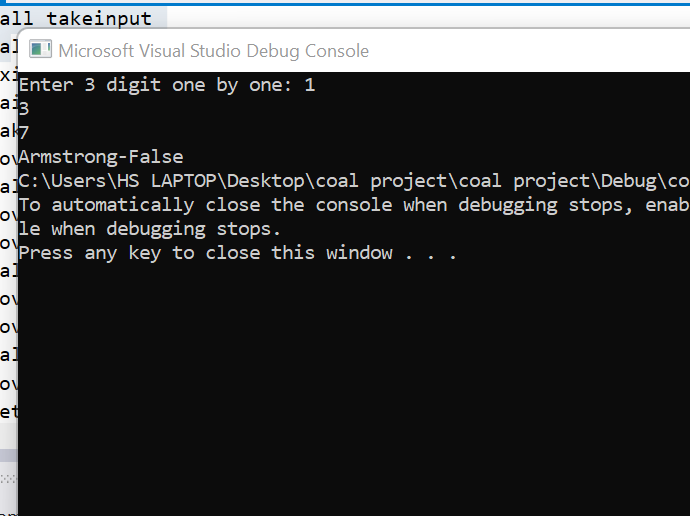
main Proc

mov edx,offset enters

call writestring

call takeinput

call Armstrong



exit

main endp

takeinput proc

mov eax,0

call readint

mov var1,eax

mov eax,0

call readint

mov var2,eax

mov eax,0

call readint

mov var3,eax

ret

takeinput endp

Armstrong proc

mov eax,var1

imul eax,var1

imul eax,var1

mov ebx,var2

imul ebx,var2

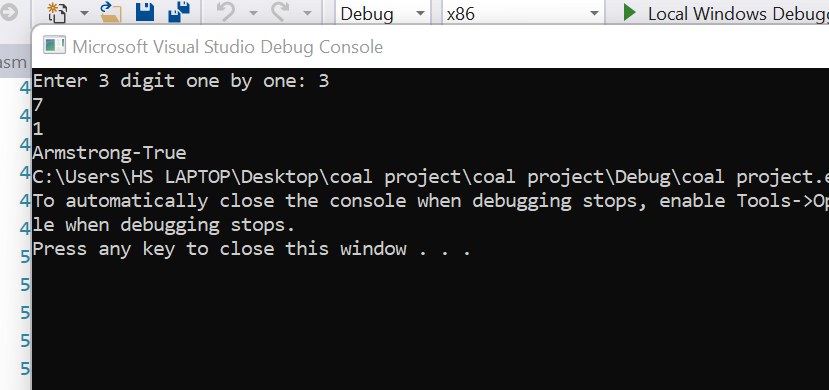
imul ebx,var2

mov ecx,var3

imul ecx,var3

imul ecx,var3

add eax,ebx



add eax,ecx

mov ebx,var1

imul ebx,100

mov ecx,var2

imul ecx,10

add var3,ebx

add var3,ecx

cmp var3,eax

jne validn

mov edx,offset valid

jmp endd

validn:

mov edx,offset invalid

endd:

call writestring

ret

Armstrong Endp

end main

**TASK 3**

INCLUDE Irvine32.inc

.data

prompt BYTE "COAL LAB",0

.code

MAIN PROC

mov esi,OFFSET prompt

mov ecx,LENGTHOF prompt

mov ebx,ecx

sub ecx, 2

shr ebx, 1

mov eax, 0

mov edx, 0

call Reversestring

mov edx, OFFSET prompt

call WriteString

exit

MAIN ENDP

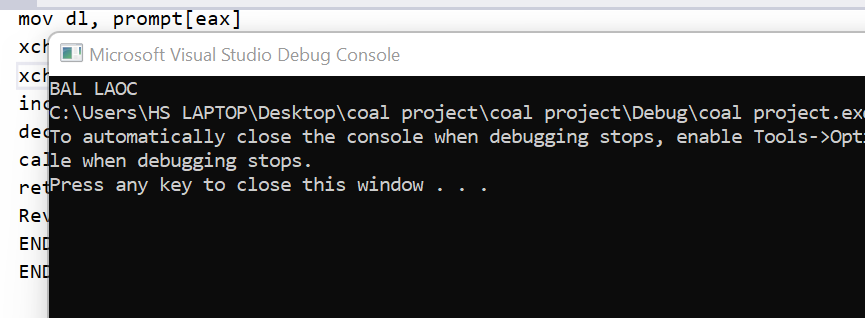
Reversestring PROC

cmp ebx, eax

jle return

mov dl, prompt[eax]

xchg dl, prompt[ecx]



xchg dl, prompt[eax]

inc eax

dec ecx

call Reversestring

return: ret

Reversestring ENDP

END MAIN

END

**TASK 4**

INCLUDE Irvine32.inc

.data

Number DWORD ?

prompt1 BYTE "Enter a number: ", 0

prompt2 BYTE "The answer is: ", 0

.code

main PROC

mov edx, OFFSET prompt1

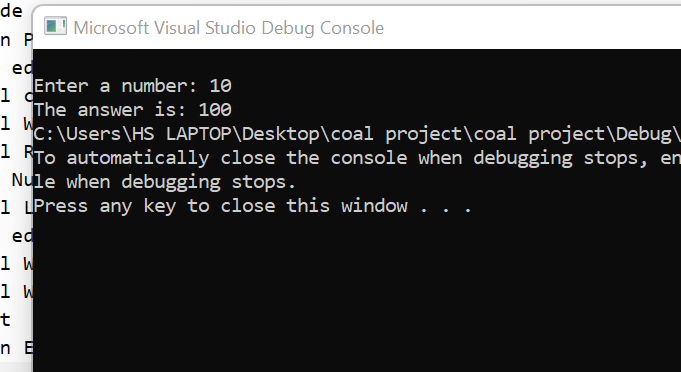
call crlf

call Writestring

call Readint

mov Number, eax

call LocalSquare



mov edx, OFFSET prompt2

call Writestring

call WriteDec

exit

main ENDP

LocalSquare PROC

Enter 69, 0

mov eax, Number

mov edx, 0

mov [ebp - 8], eax

mov ebx, [ebp - 8]

mul ebx

Leave

ret

LocalSquare ENDP

END main

**TASK 5**

INCLUDE Irvine32.inc

.data

prompt1 BYTE "Enter a number : ", 0

prompt2 BYTE "Factorial = ", 0

variable DWORD ?

.code

main PROC

mov edx, OFFSET prompt1

call Writestring

call Readint

mov variable, eax

mov ebx, variable

dec ebx

call Factorial

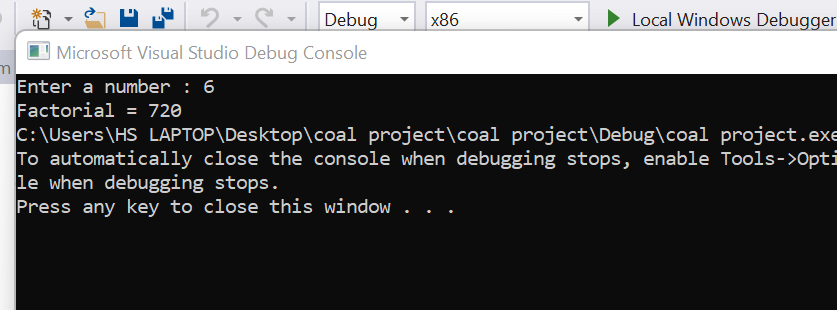
mov edx, OFFSET prompt2

call Writestring

call Writedec

exit

main ENDP



Factorial PROC

local multiplicand: DWORD

cmp ebx, 0

mov edx, 0

jz End\_Recursion

mul ebx

dec ebx

call Factorial

End\_Recursion:

ret

Factorial ENDP

end MAIN

**TASK 6**

INCLUDE Irvine32.inc

.data

prompt1 BYTE "Input : ", 0

prompt2 BYTE "Prime Number", 0

prompt3 BYTE "Not Prime Number", 0

prompt4 BYTE "Largest Prime Number: ", 0

arr DWORD 4 DUP(0)

count DWORD 0

prCount DWORD 0

.code

main PROC

mov ecx, 4

mov esi, 0

i:

mov edx, offset prompt1

call writeString

call readDec

mov arr[esi], eax

add esi, 4

loop i

push offset arr

call checkPrime

exit

main ENDP

checkPrime PROC

pop ebx

pop esi

push ebx

mov edi, 0

mov ecx, lengthof arr

j:

push ecx

mov ecx, [esi + edi \* 4]

push edi

mov edi, ecx

mov ebx, 1

mov count, 0

k:

mov eax, edi

cdq

idiv ebx

cmp edx, 0

jne el

inc count

el:

inc ebx

loop k

cmp count, 2

jne notPrime

mov edx, offset prompt2

inc prCount

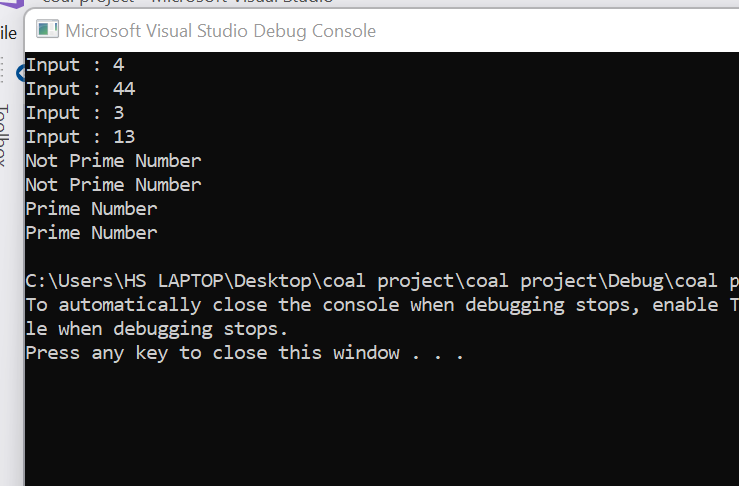
jmp endd

notPrime:

mov edx, offset prompt3

endd:

call writeString



call crlf

pop edi

inc edi

pop ecx

loop j

cmp prCount, 4

jne khtm

call largestPrime

khtm:

ret

checkPrime ENDP

largestPrime PROC

mov ecx, lengthof arr

dec ecx

mov eax, [esi]

x:

cmp eax, [esi+4]

jg next

mov eax, [esi+4]

next:

add esi, 4

loop x

mov edx, offset prompt4

call writeString

call writeDec

ret

largestPrime ENDP

END main