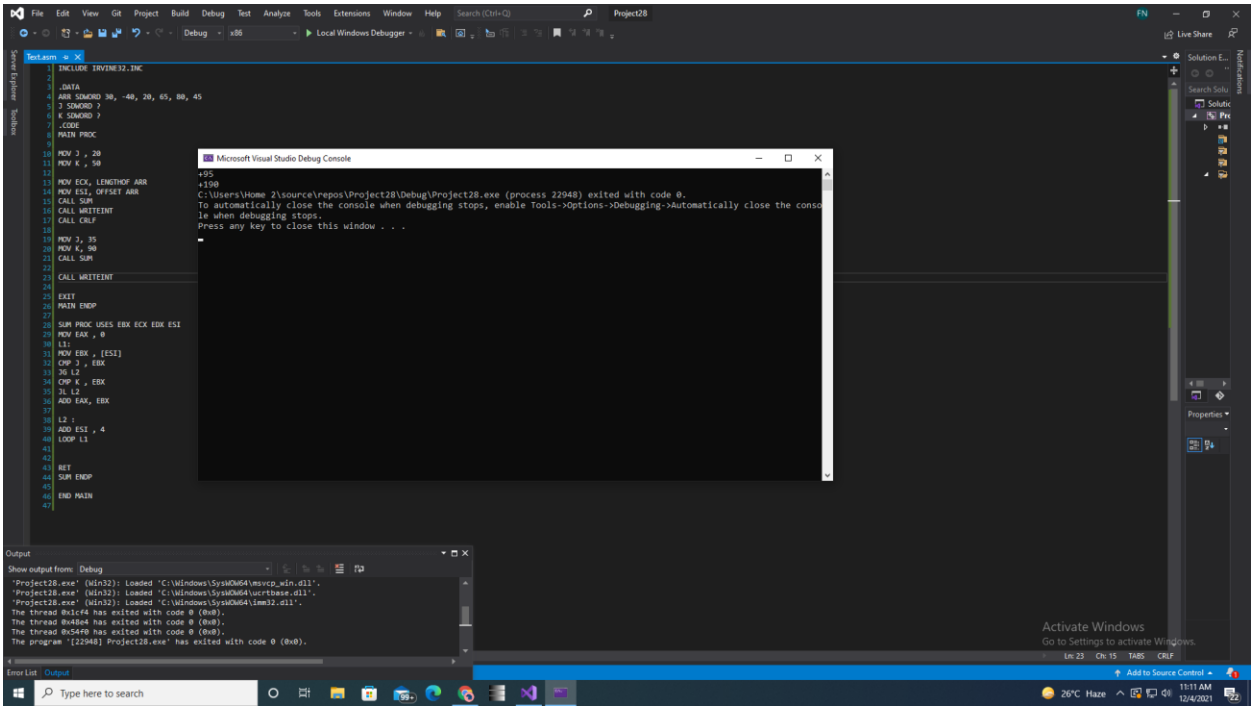
` **ASSIGNMENT NO 2**

K200368 HAMMAD SHAKEEL

Q NO 1:

RESULT:



DOCUMENTATION:

We pass the length of the array and the address of the array using the registers and call the function. Now firstly, we are preserving the values of the resisters the initializing our register EAX to 0 for adding purposes. Then we are traversing the entire array to find the number that falls within the given ranges. If it falls, then it add the number to EAX and then moving the pointer to point to the next index of the array. After traversing the entire array it simply return the function and the value of the sum through the EAX register.

CODE :

INCLUDE IRVINE32.INC

.DATA

ARR SDWORD 30, -40, 20, 65, 80, 45

J SDWORD ?

```
K SDWORD ?
.CODE
MAIN PROC

MOV J , 20
MOV K , 50

MOV ECX, LENGTHOF ARR
MOV ESI, OFFSET ARR
CALL SUM
CALL WRITEINT
CALL CRLF

MOV J, 35
MOV K, 90
CALL SUM

CALL WRITEINT

EXIT
MAIN ENDP

SUM PROC USES EBX ECX EDX ESI
MOV EAX , 0
L1:
MOV EBX , [ESI]
CMP J , EBX
JG L2
CMP K , EBX
```

JL L2

ADD EAX, EBX

L2 :

ADD ESI , 4

LOOP L1

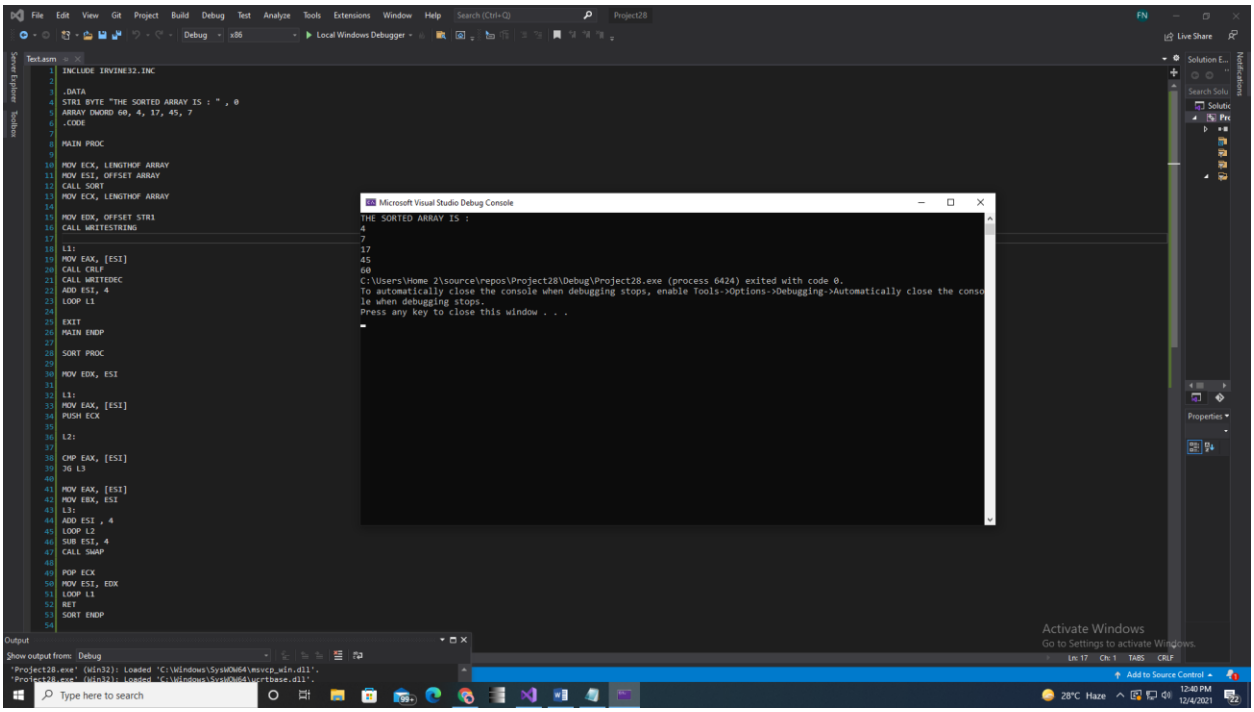
RET

SUM ENDP

END MAIN

Q NO 2:

RESULT:



CODE :

```
INCLUDE IRVINE32.INC

.DATA
STR1 BYTE "THE SORTED ARRAY IS : " , 0
ARRAY DWORD 60, 4, 17, 45, 7
.CODE

MAIN PROC

MOV ECX, LENGTHOF ARRAY
MOV ESI, OFFSET ARRAY
CALL SORT
MOV ECX, LENGTHOF ARRAY

MOV EDX, OFFSET STR1
CALL WRITESTRING

L1:
MOV EAX, [ESI]
CALL CRLF
CALL WRITEDEC
ADD ESI, 4
LOOP L1

EXIT
MAIN ENDP

SORT PROC
```

```
MOV EDX, ESI

L1:
MOV EAX, [ESI]
PUSH ECX

L2:

CMP EAX, [ESI]
JG L3

MOV EAX, [ESI]
MOV EBX, ESI
L3:
ADD ESI , 4
LOOP L2
SUB ESI, 4
CALL SWAP

POP ECX
MOV ESI, EDX
LOOP L1
RET
SORT ENDP

SWAP PROC
XCHG EAX, [ESI]
MOV ESI, EBX
```

MOV [ESI], EAX

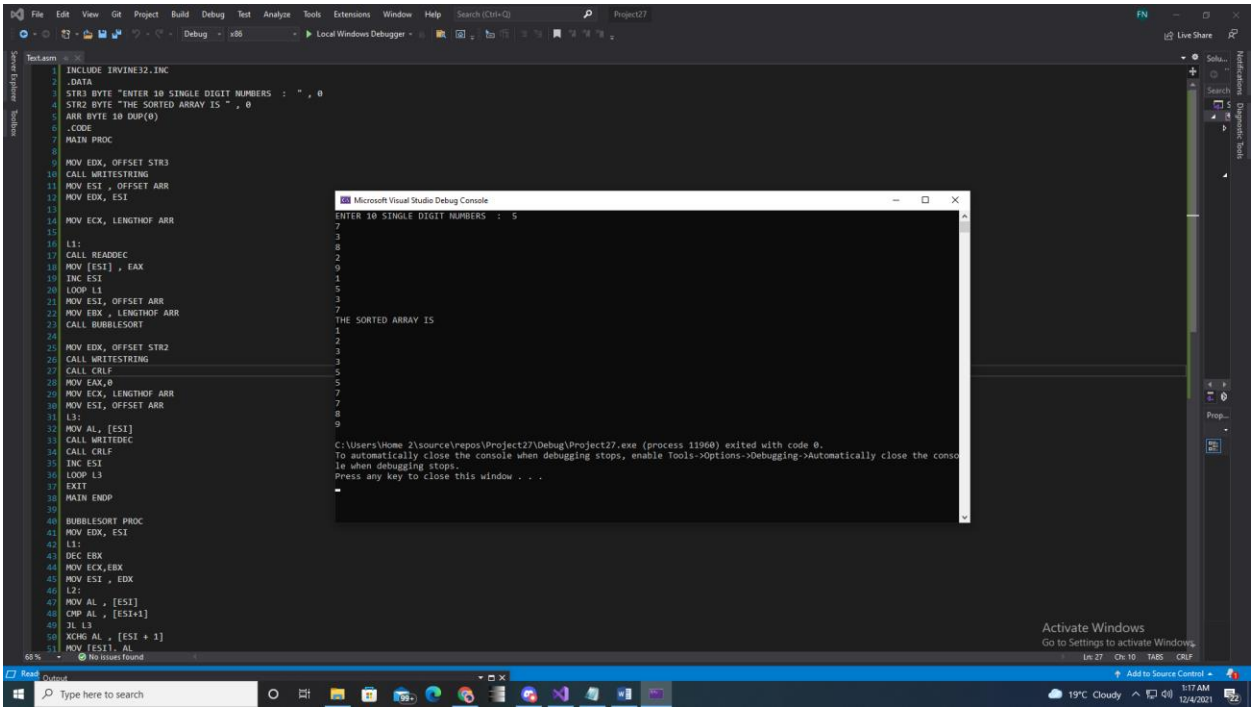
RET

SWAP ENDP

END MAIN

Q NO 3:

RESULT:



CODE :

INCLUDE IRVINE32.INC

.DATA

STR3 BYTE "ENTER 10 SINGLE DIGIT NUMBERS : " , 0

STR2 BYTE "THE SORTED ARRAY IS " , 0

ARR BYTE 10 DUP(0)

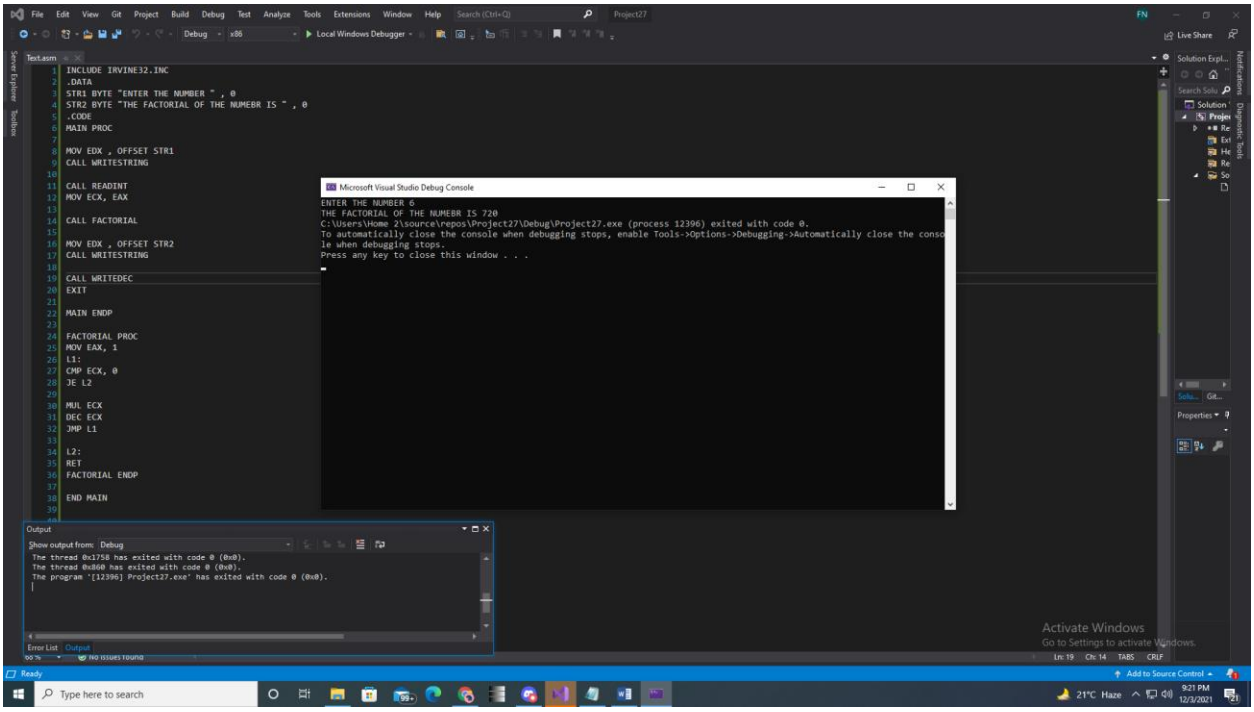
.CODE

```
MAIN PROC

MOV EDX, OFFSET STR3
CALL WRITESTRING
MOV ESI , OFFSET ARR
MOV EDX, ESI

MOV ECX, LENGTHOF ARR

L1:
CALL READDEC
MOV [ESI] , EAX
INC ESI
LOOP L1
MOV ESI, OFFSET ARR
MOV EBX , LENGTHOF ARR
CALL BUBBLESORT

MOV EDX, OFFSET STR2
CALL WRITESTRING
CALL CRLF
MOV EAX,0
MOV ECX, LENGTHOF ARR
MOV ESI, OFFSET ARR
L3:
MOV AL, [ESI]
CALL WRITEDEC
CALL CRLF
INC ESI
```

```
LOOP L3
EXIT
MAIN ENDP

BUBBLESORT PROC
MOV EDX, ESI
L1:
DEC EBX
MOV ECX,EBX
MOV ESI , EDX
L2:
MOV AL , [ESI]
CMP AL , [ESI+1]
JL L3
XCHG AL , [ESI + 1]
MOV [ESI], AL

L3:
INC ESI
LOOP L2
MOV ECX,EBX
MOV ECX,EBX
LOOP L1
RET
BUBBLESORT ENDP

END MAIN
```

Q NO 4:

RESULT:



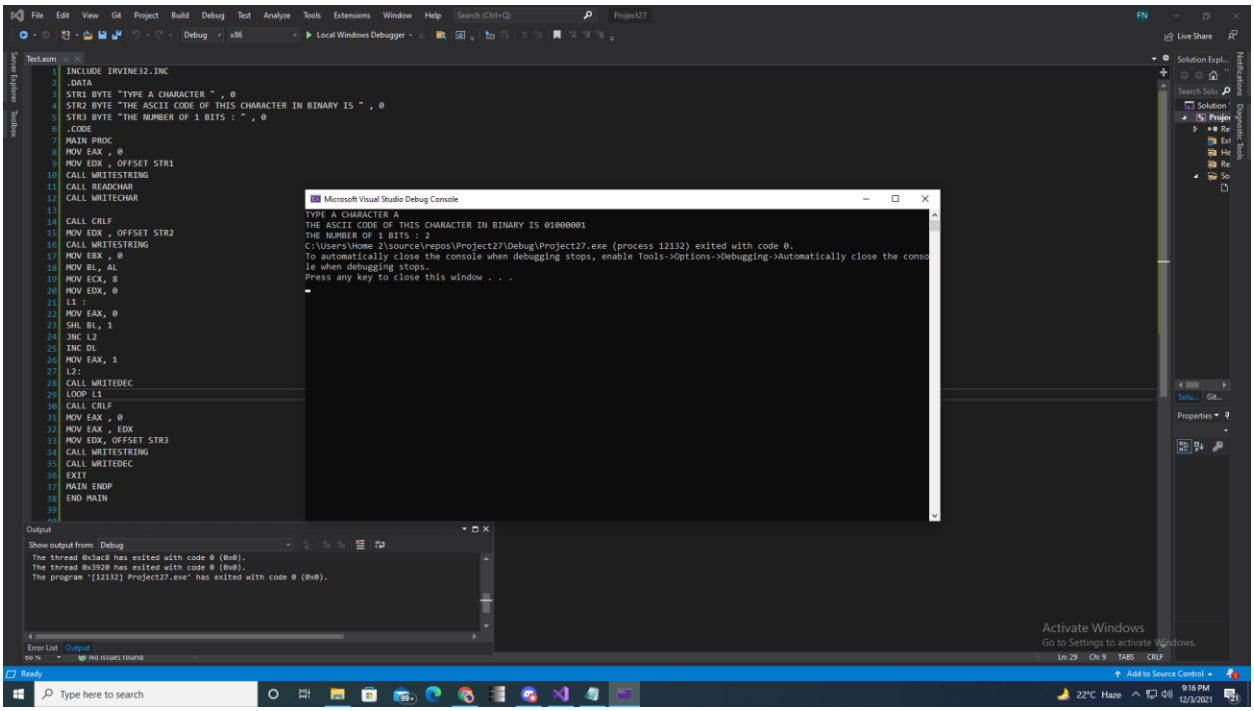
CODE :

```
INCLUDE IRVINE32.INC
.DATA
STR1 BYTE "ENTER THE NUMBER " , 0
STR2 BYTE "THE FACTORIAL OF THE NUMEBR IS " , 0
.CODE
MAIN PROC

MOV EDX , OFFSET STR1
CALL WRITESTRING

CALL READINT
MOV ECX, EAX
```

```
CALL FACTORIAL

MOV EDX , OFFSET STR2
CALL WRITESTRING

CALL WRITEINT
EXIT

MAIN ENDP

FACTORIAL PROC
MOV EAX, 1
L1:
CMP ECX, 0
JE L2

MUL ECX
DEC ECX
JMP L1

L2:
RET
FACTORIAL ENDP

END MAIN
```

Q NO 5:

RESULT:



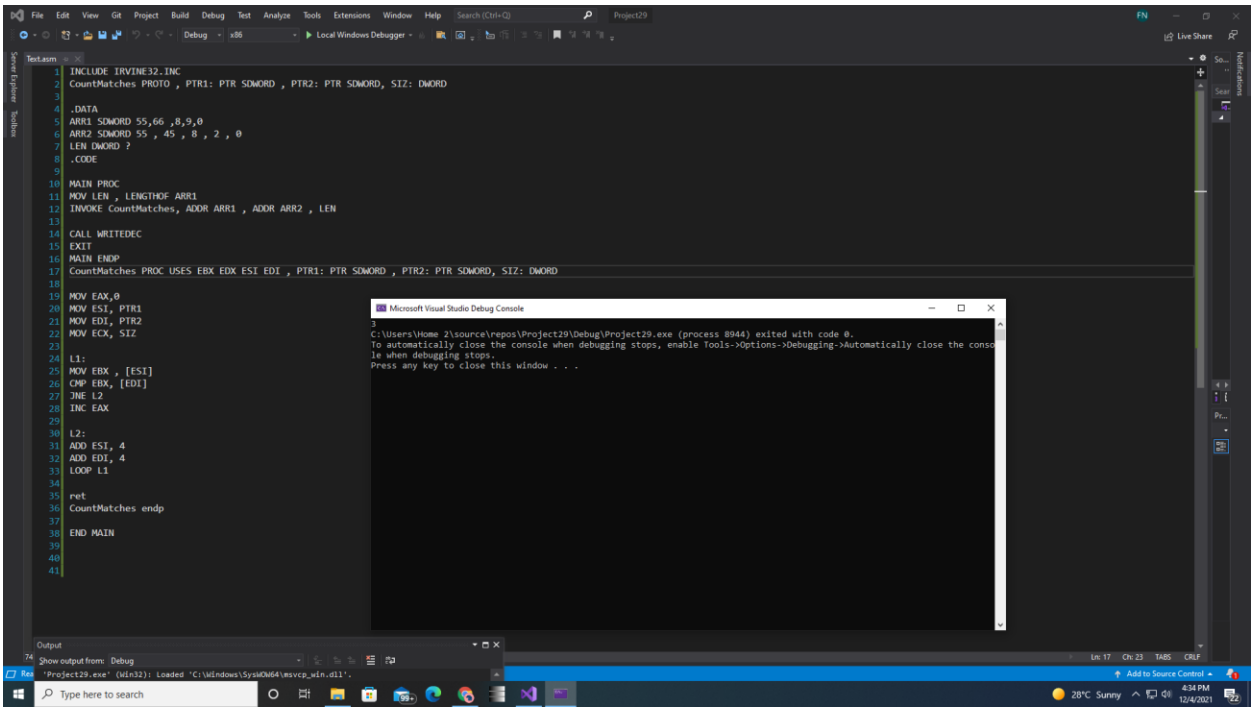
CODE :

```
INCLUDE IRVINE32.INC
.DATA
STR1 BYTE "TYPE A CHARACTER " , 0
STR2 BYTE "THE ASCII CODE OF THIS CHARACTER IN BINARY IS " , 0
STR3 BYTE "THE NUMBER OF 1 BITS : " , 0
.CODE
MAIN PROC
MOV EAX , 0
MOV EDX , OFFSET STR1
CALL WRITESTRING
CALL READCHAR
CALL WRITECHAR

CALL CRLF
```

```
MOV EDX , OFFSET STR2
CALL WRITESTRING
MOV EBX , 0
MOV BL, AL
MOV ECX, 8
MOV EDX, 0
L1 :
MOV EAX, 0
SHL BL, 1
JNC L2
INC DL
MOV EAX, 1
L2:
CALL WRITEDEC
LOOP L1
CALL CRLF
MOV EAX , 0
MOV EAX , EDX
MOV EDX, OFFSET STR3
CALL WRITESTRING
CALL WRITEDEC
EXIT
MAIN ENDP
END MAIN
```

Q NO 6:

RESULT:



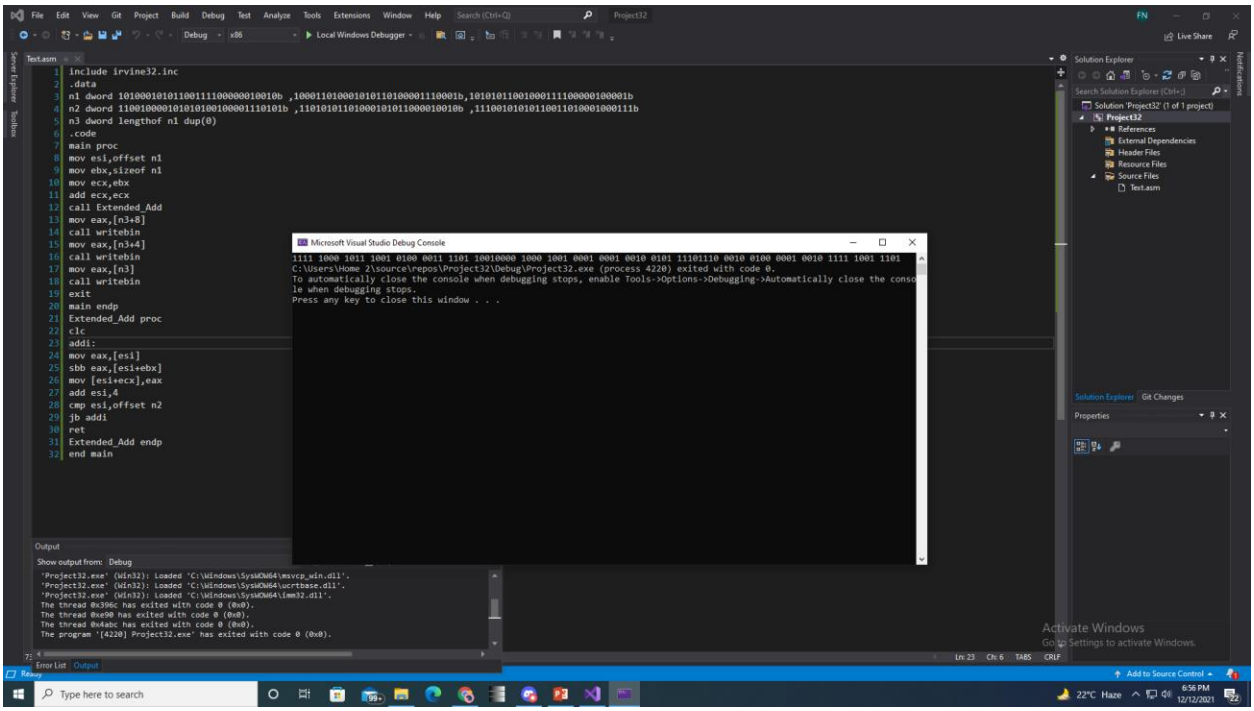
CODE :

```
INCLUDE IRVINE32.INC
CountMatches PROTO , PTR1: PTR SDWORD , PTR2: PTR SDWORD, SIZ: DWORD

.DATA
ARR1 SDWORD 55,66 ,8,9,0
ARR2 SDWORD 55 , 45 , 8 , 2 , 0
LEN DWORD ?
.CODE

MAIN PROC
MOV LEN , LENGTHOF ARR1
INVOKE CountMatches, ADDR ARR1 , ADDR ARR2 , LEN

CALL WRITEDEC
EXIT
```

```
MAIN ENDP
CountMatches PROC USES EBX EDX ESI EDI , PTR1: PTR SDWORD , PTR2: PTR SDWORD, SIZ: DWORD

MOV EAX,0
MOV ESI, PTR1
MOV EDI, PTR2
MOV ECX, SIZ

L1:
MOV EBX , [ESI]
CMP EBX, [EDI]
JNE L2
INC EAX

L2:
ADD ESI, 4
ADD EDI, 4
LOOP L1

ret
CountMatches endp

END MAIN
```

Q NO 7:

RESULT:



CODE:

include irvine32.inc

.data

n1 dword 10100010101100111100000010010b
,1000110100010101101000011100001b,1010101100100011110000010000 1b

n2 dword 1100100001010101001000011101 01b ,1101010111000101011000010010b
,1110010101011001101000100011 1b

n3 dword lengthof n1 dup(0)

.code

main proc

mov esi,offset n1

mov ebx,sizeof n1

mov ecx,ebx

add ecx,ecx

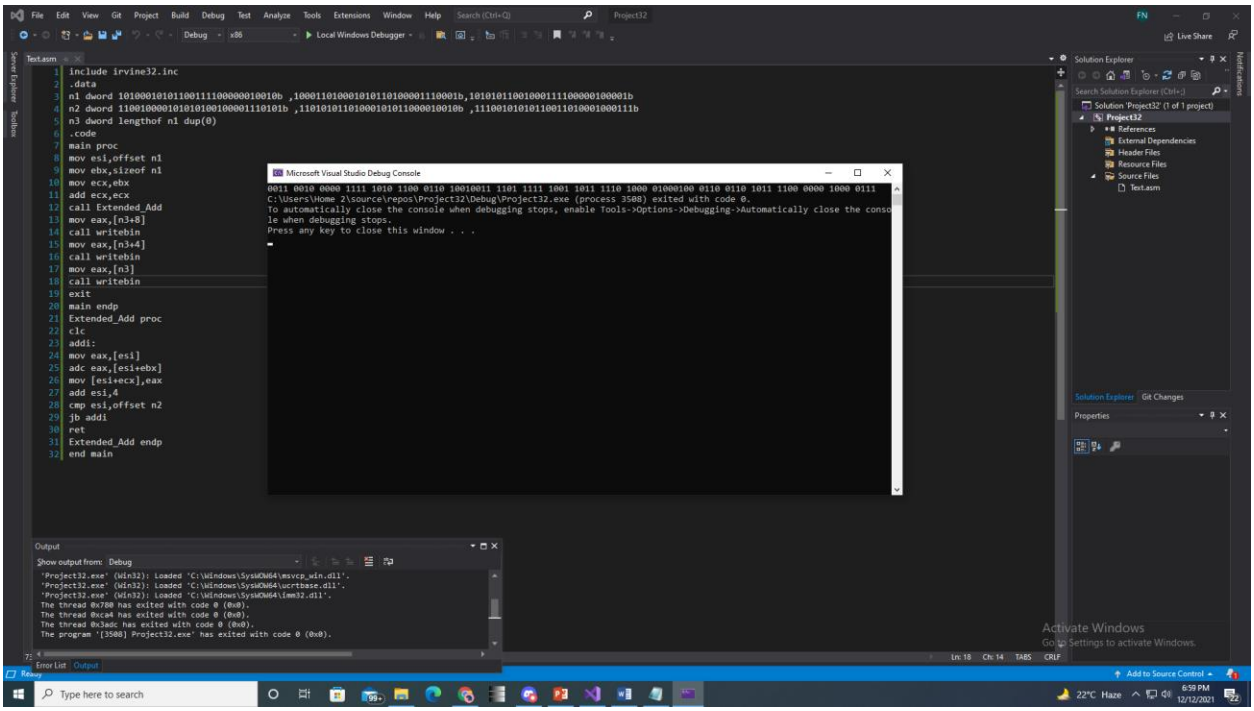
call Extended_Add

mov eax,[n3+8]

```
call writebin
mov eax,[n3+4]
call writebin
mov eax,[n3]
call writebin
exit
main endp
Extended_Add proc
clc
addi:
mov eax,[esi]
sbb eax,[esi+ebx]
mov [esi+ecx],eax
add esi,4
cmp esi,offset n2
jb addi
ret
Extended_Add endp
end main
```

Q NO 8:

RESULT:



CODE:

include irvine32.inc

.data

n1 dword 10100010101100111100000010010b
,1000110100010101101000011100001b,1010101100100011110000010000 1b

n2 dword 1100100001010101001000011101 01b ,1101010110100010101100001001 0b
,1110010101011001100010000111b

n3 dword lengthof n1 dup(0)

.code

main proc

mov esi,offset n1

mov ebx,sizeof n1

mov ecx,ebx

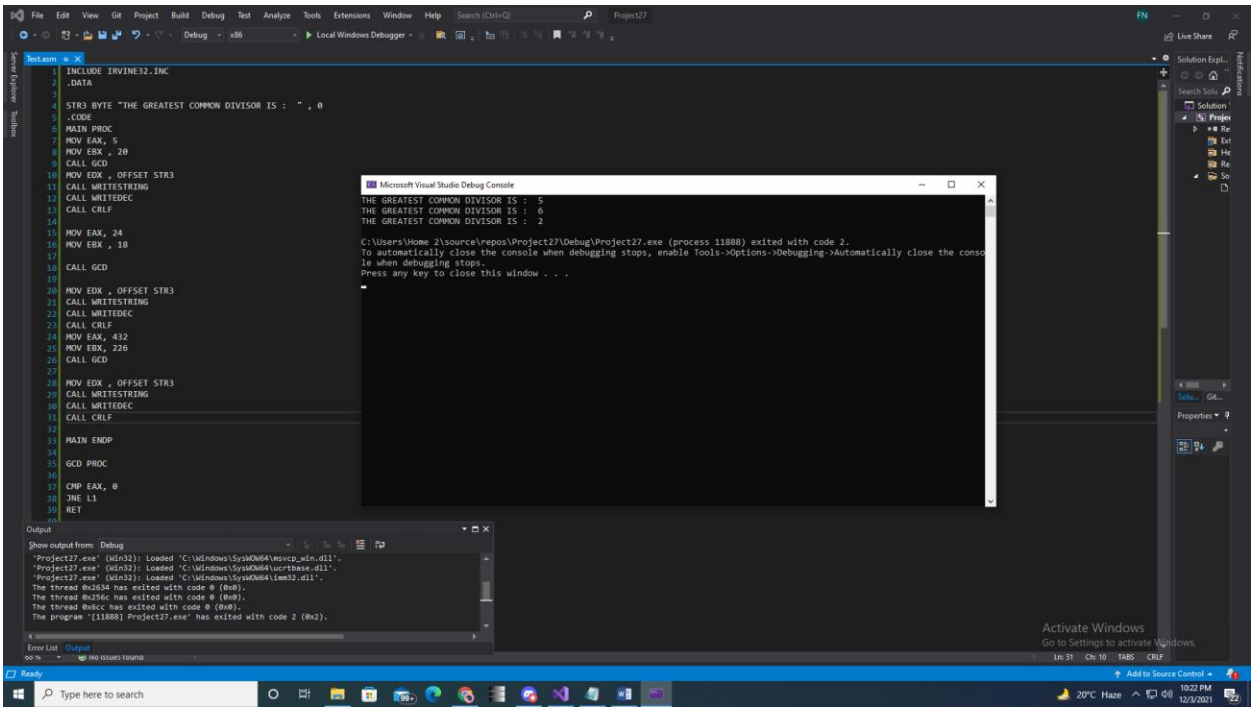
add ecx,ecx

call Extended_Add

```
mov eax,[n3+8]
call writebin
mov eax,[n3+4]
call writebin
mov eax,[n3]
call writebin
exit
main endp
Extended_Add proc
clc
addi:
mov eax,[esi]
adc eax,[esi+ebx]
mov [esi+ecx],eax
add esi,4
cmp esi,offset n2
jb addi
ret
Extended_Add endp
end main
```

Q NO 9:

RESULT:



CODE :

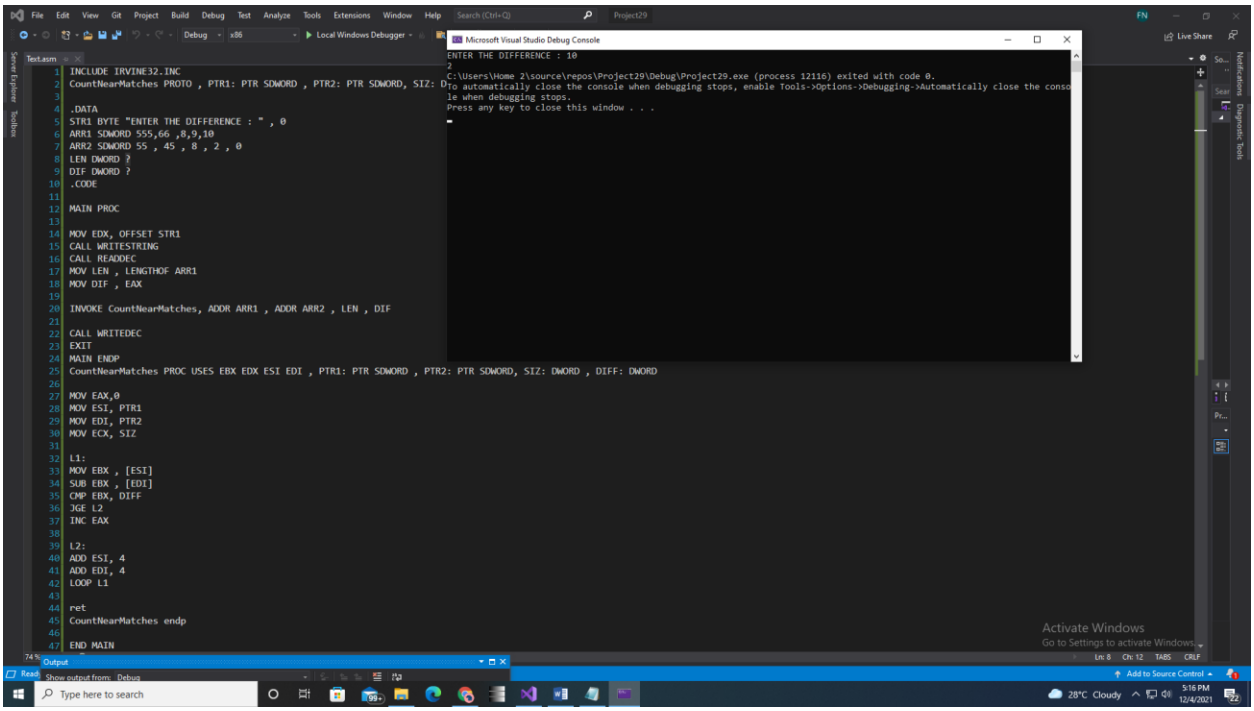
```
INCLUDE IRVINE32.INC
.DATA

STR3 BYTE "THE GREATEST COMMON DIVISOR IS : " , 0
.CODE
MAIN PROC
MOV EAX, 5
MOV EBX , 20
CALL GCD
MOV EDX , OFFSET STR3
CALL WRITESTRING
CALL WRITEDEC
CALL CRLF

MOV EAX, 24
```

```
MOV EBX , 18

CALL GCD

MOV EDX , OFFSET STR3
CALL WRITESTRING
CALL WRITEDEC
CALL CRLF
MOV EAX, 432
MOV EBX, 226
CALL GCD

MOV EDX , OFFSET STR3
CALL WRITESTRING
CALL WRITEDEC
CALL CRLF

MAIN ENDP

GCD PROC

CMP EAX, 0
JNE L1
RET

L1:
CMP EBX, 0
JNE L2
RET
```

```
L2:

CMP EAX , EBX
JNE L3
RET

L3:

CMP EAX , EBX
JNG L4
SUB EAX , EBX
CALL GCD
RET
L4:
SUB EBX , EAX
CALL GCD
RET
GCD ENDP
END MAIN
```

Q NO 10

RESULT:



CODE:

```
INCLUDE IRVINE32.INC
CountNearMatches PROTO , PTR1: PTR SDWORD , PTR2: PTR SDWORD, SIZ: DWORD , DIFF: DWORD

.DATA
STR1 BYTE "ENTER THE DIFFERENCE : " , 0
ARR1 SDWORD 555,66 ,8,9,10
ARR2 SDWORD 55 , 45 , 8 , 2 , 0
LEN DWORD ?
DIF DWORD ?
.CODE

MAIN PROC

MOV EDX, OFFSET STR1
CALL WRITESTRING
```

```
CALL READDEC
MOV LEN , LENGTHOF ARR1
MOV DIF , EAX

INVOKE CountNearMatches, ADDR ARR1 , ADDR ARR2 , LEN , DIF

CALL WRITEDEC
EXIT
MAIN ENDP
CountNearMatches PROC USES EBX EDX ESI EDI , PTR1: PTR SDWORD , PTR2: PTR SDWORD, SIZ: DWORD
, DIFF: DWORD

MOV EAX,0
MOV ESI, PTR1
MOV EDI, PTR2
MOV ECX, SIZ

L1:
MOV EBX , [ESI]
SUB EBX , [EDI]
CMP EBX, DIFF
JGE L2
INC EAX

L2:
ADD ESI, 4
ADD EDI, 4
LOOP L1
```

```
ret

CountNearMatches endp


END MAIN
```