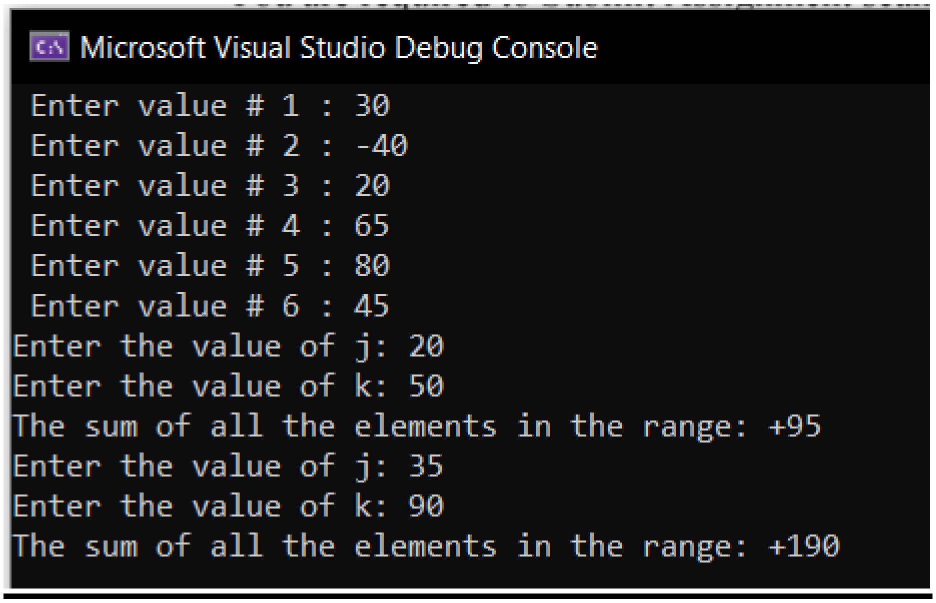
QUESTION # 1:

Code:

```
TITLE 20K-0208-Q1
Include Irvine32.inc
Include Macros.inc
.data
     arr SDWORD 6 DUP(?)
     j SDWORD ?
     k SDWORD ?
.code
     MAIN PROC
          mov ecx,LENGTHOF arr
          mov ebx,0
          getInput:
               mWrite " Enter value # "
               mov eax,ebx
               inc eax
               call WriteDec
               mWrite " : "
               call ReadInt
               mov [arr+ebx*4],eax
               inc ebx
          loop getInput
          call getJK
          push OFFSET arr
          push SIZEOF arr
          push j
          push k
          call arraySum
          call printResults
          call getJK
          push OFFSET arr
          push SIZEOF arr
          push j
          push k
          call arraySum
          call printResults
          exit
     MAIN ENDP

     printResults PROC
          mWrite "The sum of all the elements in the range: "
          call WriteInt
```

```
      call crlf
      ret
printResults ENDP

getJK PROC
      mWrite "Enter the value of j: "
      call ReadInt
      mov j,eax
      mWrite "Enter the value of k: "
      call ReadInt
      mov k,eax
      ret
getJK ENDP

arraySum PROC uses EBX ECX EDX ESI
      local first:SDWORD,last:SDWORD,sizeArray:DWORD
      mov esi,[ebp + 20]
      mov eax,[ebp + 16]
      mov sizeArray,eax
      mov eax,[ebp + 12]
      mov first,eax
      mov eax,[ebp + 8]
      mov last,eax
      mov eax,0
      mov edx,0
      mov ecx,sizeArray
      sumInRange:
            mov ebx,[esi + edx * 4]
            cmp ebx,first
            jge checkIfWithinRange
            jmp continueLoop
            checkIfWithinRange:
                  cmp ebx,last
                  jle addIt
                  jmp continueLoop
            addIt:
                  add eax,ebx
            continueLoop:
                  inc ecx
                  sub ecx,4
                  inc edx
      loop sumInRange
      ret 16
arraySum ENDP
END MAIN
```

Output:



```
 Enter value # 1 : 30
 Enter value # 2 : -40
 Enter value # 3 : 20
 Enter value # 4 : 65
 Enter value # 5 : 80
 Enter value # 6 : 45
Enter the value of j: 20
Enter the value of k: 50
The sum of all the elements in the range: +95
Enter the value of j: 35
Enter the value of k: 90
The sum of all the elements in the range: +190
```

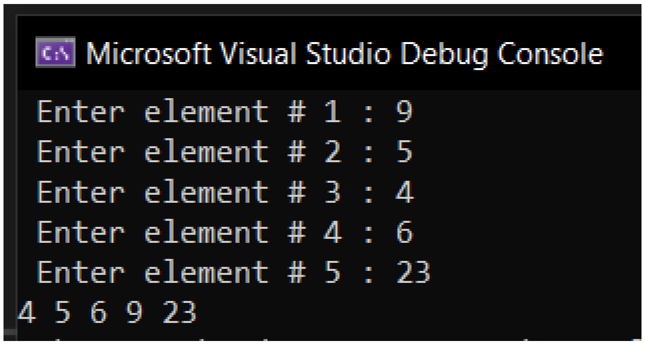
QUESTION # 2:

Code:

```
TITLE 20K-0208-Q2
Include Irvine32.inc
Include Macros.inc
.data
     arr DWORD 5 DUP(?)
.code
     MAIN PROC
          mov ecx,LENGTHOF arr
          mov ebx,0
          getInput:
               mWrite " Enter element # "
               mov eax,ebx
               inc eax
               call WriteDec
               mWrite " : "
               call ReadInt
               mov [arr+ebx*4],eax
               inc ebx
          loop getInput
          call selectionSort
          mov ecx,LENGTHOF arr
          mov ebx,0
          printSortedArray:
               mov eax,[arr + ebx * 4]
               call WriteDec
               mWrite " "
               inc ebx
          loop printSortedArray
          exit
     MAIN ENDP

     SWAP PROC
          push ebp
          mov ebp,esp
          mov edx,[ebp + 8]
          push edx
          mov eax,[arr + edx * 4]
          mov edx,[ebp + 12]
          xchg eax,[arr + edx * 4]
          pop edx
          mov [arr + edx * 4],eax
```

```
      pop ebp
      ret 8
SWAP ENDP

selectionSort PROC
      LOCAL largest:DWORD,i:DWORD,j:DWORD
      mov ecx,LENGTHOF arr
      mov largest,0
      dec ecx
      mov i,ecx
      mov j,ecx
      outerLoop:
            mov ebx,i
            mov largest,ebx
            push ecx
            mov edx,i
            mov j,edx
            innerLoop:
                  dec j
                  mov edx,j
                  mov eax,[arr + edx * 4]
                  mov edx,largest
                  mov ebx,[arr + edx * 4]
                  cmp eax,ebx
                  jg markNewMax
                  jmp continueLoop
                  markNewMax:
                        mov edx,j
                        mov largest,edx
                  continueLoop:
            loop innerLoop
            push i
            push largest
            call SWAP
            pop ecx
            dec i
      loop outerLoop
      ret
selectionSort ENDP
END MAIN
```

Output:



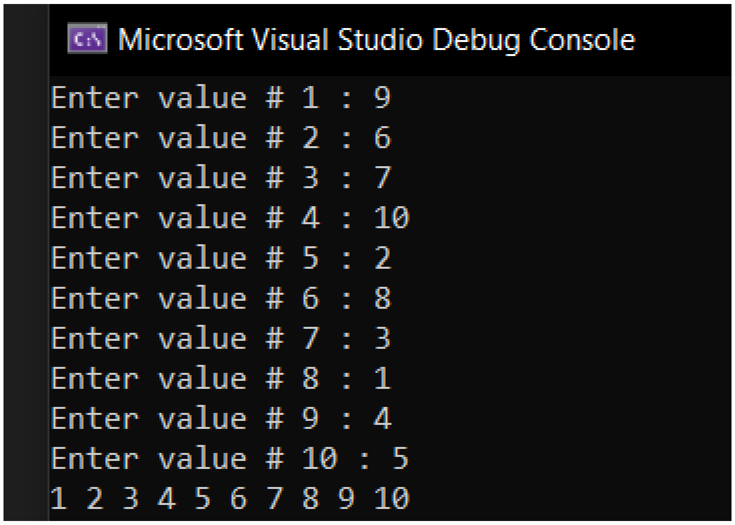
```
 Enter element # 1 : 9
 Enter element # 2 : 5
 Enter element # 3 : 4
 Enter element # 4 : 6
 Enter element # 5 : 23
4 5 6 9 23
```

QUESTION # 3:

Code:

```
TITLE 20K-0208-Q3
Include Irvine32.inc
Include Macros.inc
.data
      arr BYTE 10 DUP(?)
.code
      MAIN PROC
            mov ecx,LENGTHOF arr
            mov ebx,0
            getInput:
                  mWrite "Enter value # "
                  mov eax,ebx
                  inc eax
                  call WriteDec
                  mWrite " : "
                  call ReadInt
                  mov [arr + ebx],al
                  inc ebx
            loop getInput
            mov esi,OFFSET arr
            mov ebx,LENGTHOF arr
            call bubbleSort
            mov ebx,0
            mov ecx,LENGTHOF arr
            printArray:
                  mov al,[arr + ebx]
                  call WriteDec
                  inc ebx
                  mWrite " "
            loop printArray
            exit
      MAIN ENDP

      bubbleSort PROC
            mov edi,esi
            mov ecx,ebx
            dec ecx
            mov ebx,0
            mov eax,0
            outerLoop:
                  push ecx
            mov esi,edi
            innerLoop:
                mov al,[esi]
                mov bl,[esi + 1]
                cmp al,bl
                jg swapElements
                continueLoop:
                    inc esi
            loop innerLoop
            pop ecx
        loop outerLoop
        jmp endProgram
        swapElements:
            mov al,[esi]
            mov bl,[esi + 1]
            xchg al,bl
            mov [esi],al
            mov [esi + 1],bl
            jmp continueLoop
        endProgram:
        ret 8
    bubbleSort ENDP
    END MAIN
```

Output:



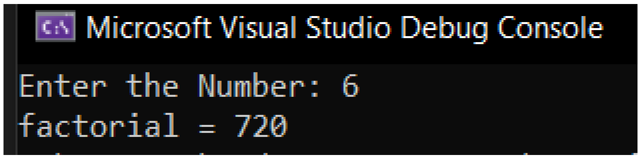
```
Enter value # 1 : 9
Enter value # 2 : 6
Enter value # 3 : 7
Enter value # 4 : 10
Enter value # 5 : 2
Enter value # 6 : 8
Enter value # 7 : 3
Enter value # 8 : 1
Enter value # 9 : 4
Enter value # 10 : 5
1 2 3 4 5 6 7 8 9 10
```

QUESTION # 4:

Code:

```
TITLE 20K-0208-Q4
Include Irvine32.inc
Include Macros.inc
.data
     N DWORD ?
.code
     MAIN PROC
          mWrite "Enter the Number: "
          call ReadInt
          mov N,eax
          call factorial
          mWrite "factorial = "
          call WriteDec
          exit
     MAIN ENDP

     factorial PROC
          mov eax,1
          cmp N,0
          jle endProgram
          mov ecx,N
          calculate:
               mov edx,0
               mul ecx
          loop calculate
          endProgram:
          ret
     factorial ENDP
     END MAIN
```

Output:



```
Enter the Number: 6
factorial = 720
```

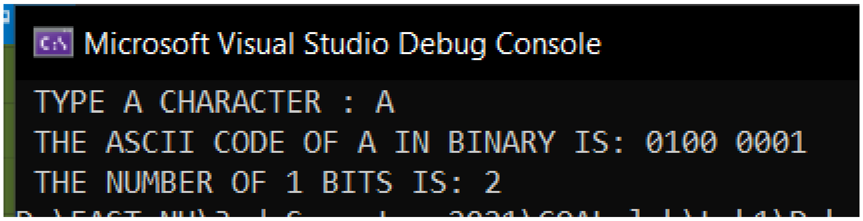
QUESTION # 5:

Code:

```
TITLE 20K-0208-Q5
Include Irvine32.inc
Include Macros.inc
.data
     character BYTE ?
     binaryCode DWORD 00000000b
     oneCount DWORD 0
.code
     MAIN PROC
          mov eax,0
          mov edx,0
          mWrite " TYPE A CHARACTER : "
          call ReadChar
          call WriteChar
          mov character,al
          mov ah,0
          movzx ebx,al
          mov binaryCode,ebx
          mov eax,binaryCode
          call countOnes
          call crlf
          mWrite " THE ASCII CODE OF "
          mov al,character
          call WriteChar
          mWrite " IN BINARY IS: "
          mov eax,binaryCode
          mov ebx,1
          call WriteBinB
          call crlf
          mWrite " THE NUMBER OF 1 BITS IS: "
          mov eax,oneCount
          call WriteDec
          exit
     MAIN ENDP

     countOnes PROC
          mov eax,binaryCode
          mov ebx,2
          loopForCount:
               mov edx,0
               div ebx
               cmp edx,1
```

```
                jz incrementCount
                cmp eax,0
                jz endLoop
                jmp continueLoop
                incrementCount:
                    inc oneCount
                continueLoop:
            loop loopForCount
            endLoop:
            ret
        countOnes ENDP
        END MAIN
```

Output:



```
TYPE A CHARACTER : A
THE ASCII CODE OF A IN BINARY IS: 0100 0001
THE NUMBER OF 1 BITS IS: 2
```

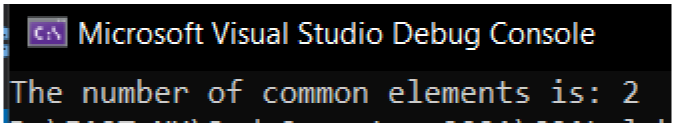
QUESTION # 6:

Code:

```
TITLE 20K-0208-Q6
Include Irvine32.inc
Include Macros.inc

countMatches PROTO,
      Parr1:PTR SDWORD,
      Parr2:PTR SDWORD,
      lengthArr:DWORD

.data
      arr1 SDWORD 1,2,3,4,5,6
      arr2 SDWORD 2,1,3,4,6,5
.code
      MAIN PROC
            INVOKE countMatches,ADDR arr1,ADDR arr2,LENGTHOF arr1
            mWrite "The number of common elements is: "
            call WriteDec
            exit
      MAIN ENDP

      countMatches PROC uses EBX ECX EDX ESI EDI,
            Parr1:PTR SDWORD,
            Parr2:PTR SDWORD,
            lengthArr:DWORD
            mov esi,Parr1
            mov edi,Parr2
            mov ebx,0
            mov eax,0
            mov ecx,lengthArr
            dec ecx
            compareElements:
                  mov edx,[esi + ebx * 4]
                  cmp edx,[edi + ebx * 4]
                  jz incrementCount
                  jmp continueLoop
                  incrementCount:
                        inc eax
                  continueLoop:
                        inc ebx
            loop compareElements
            ret
      countMatches ENDP
```

```
    END MAIN
```

Output:



```
The number of common elements is: 2
```

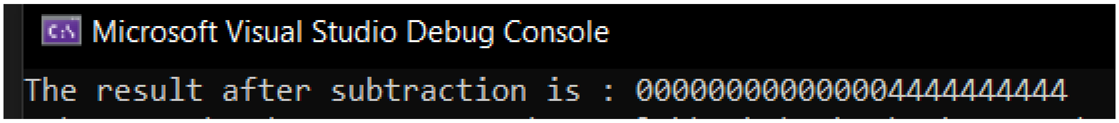
QUESTION # 7:

Code:

```
TITLE 20K-0208-Q8

Include Irvine32.inc
Include Macros.inc

.data
     num1 QWORD 0000005555666677h
     num2 QWORD 0000001111222233h
     difference DWORD 3 DUP(?)
.code
     MAIN PROC
          mov esi,OFFSET num1
          mov edi,OFFSET num2
          mov ebx,OFFSET difference
          mov ecx,2
          call Extended_Sub
          mov ecx,LENGTHOF difference
          mov ebx,ecx
          dec ebx
          mWrite "The result after subtraction is : "
          printArray:
               mov eax,[difference + ebx * 4]
               call WriteHex
               dec ebx
          loop printArray
          exit
     MAIN ENDP

     Extended_Sub PROC
          clc
          subtract:
               mov eax,[esi]
               sbb eax,[edi]
               pushfd
               mov [ebx],eax
               add esi,4
               add edi,4
               add ebx,4
               popfd
          loop subtract
          sbb WORD PTR [ebx],0
          ret
```

```
Extended_Sub ENDP
END MAIN
```

Output:



```
The result after subtraction is : 000000000000004444444444
```

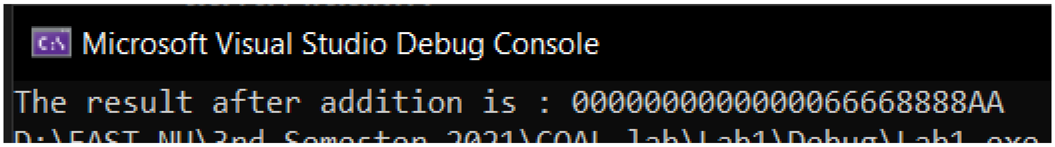
QUESTION # 8:

Code:

```
TITLE 20K-0208-Q8

Include Irvine32.inc
Include Macros.inc

.data
     num1 QWORD 0000005555666677h
     num2 QWORD 0000001111222233h
     sum DWORD 3 DUP(?)
.code
     MAIN PROC
          mov esi,OFFSET num1
          mov edi,OFFSET num2
          mov ebx,OFFSET sum
          mov ecx,2
          call Extended_Add
          mov ecx,LENGTHOF sum
          mov ebx,ecx
          dec ebx
          mWrite "The result after addition is : "
          printArray:
               mov eax,[sum + ebx * 4]
               call WriteHex
               dec ebx
          loop printArray
          exit
     MAIN ENDP

     Extended_Add PROC
          clc
          addition:
               mov eax,[esi]
               adc eax,[edi]
               pushfd
               mov [ebx],eax
               add esi,4
               add edi,4
               add ebx,4
               popfd
          loop addition
          adc WORD PTR [ebx],0
          ret
```

```
Extended_Add ENDP
END MAIN
```

Output:



```
The result after addition is : 0000000000000066668888AA
D:\FAST NU\3rd Semester 2021\COAL lab\Lab1\Debug\Lab1.exe
```

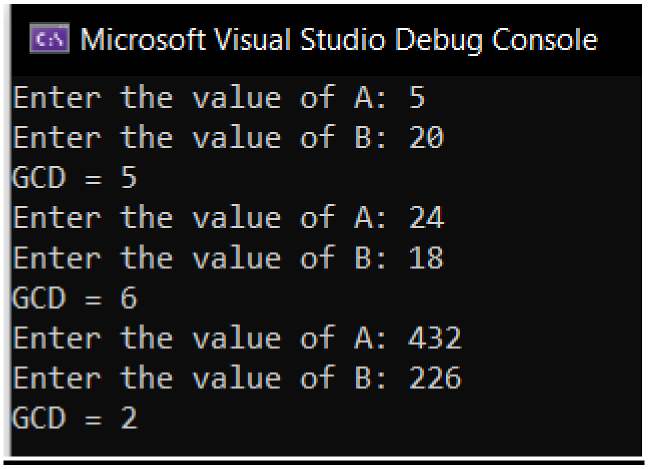
QUESTION # 9:

Code:

```
TITLE 20K-0208-Q9
Include Irvine32.inc
Include Macros.inc
.data
     A DWORD ?
     B DWORD ?
     GCD DWORD ?
.code
     MAIN PROC
          call getAB
          call getGCD
          call printResults
          call getAB
          call getGCD
          call printResults
          call getAB
          call getGCD
          call printResults
          exit
     MAIN ENDP

     printResults PROC
          mWrite "GCD = "
          mov eax,GCD
          call WriteDec
          call crlf
          ret
     printResults ENDP

     getAB PROC
          mWrite "Enter the value of A: "
          call ReadInt
          mov A,eax
          mWrite "Enter the value of B: "
          call ReadInt
          mov B,eax
          ret
     getAB ENDP

     getGCD PROC
          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
        mov GCD,eax
        ret
        jmp endProgram
    secondEnd:
        mov eax,A
        mov GCD,eax
        ret
        jmp endProgram
    thirdEnd:
        mov eax,A
        mov GCD,eax
        ret
        jmp endProgram
    firstRecurse:
        mov eax,A
        sub eax,B
        mov A,eax
        call getGCD
        jmp endProgram
    secondRecurse:
        mov eax,B
        sub eax,A
        mov B,eax
        call getGCD
    endProgram:
    ret
getGCD ENDP
END MAIN
```

Output:



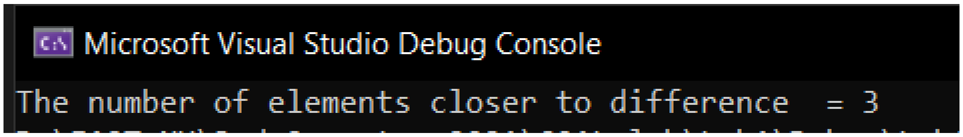
```
Enter the value of A: 5
Enter the value of B: 20
GCD = 5
Enter the value of A: 24
Enter the value of B: 18
GCD = 6
Enter the value of A: 432
Enter the value of B: 226
GCD = 2
```

QUESTION # 10:

Code:

```
TITLE 20K-0208-Q10

Include Irvine32.inc
Include Macros.inc

CountNearMatches PROTO,
      pArr1:PTR SDWORD,
      pArr2:PTR SDWORD,
      lengthArr:DWORD,
      maxDiff:SDWORD

.data
      arr1 SDWORD 10,20,30,40,50
      arr2 SDWORD 5,16,28,31,49
.code
      MAIN PROC
            INVOKE CountNearMatches, ADDR arr1, ADDR arr2,LENGTHOF arr1,
4
            mWrite "The number of elements closer to difference  = "
            call WriteDec
            exit
      MAIN ENDP

      CountNearMatches PROC USES EBX ECX EDX ESI EDI,
            pArr1:PTR SDWORD,
            pArr2:PTR SDWORD,
            lengthArr:DWORD,
            maxDiff:SDWORD

            mov eax,0
            mov esi,pArr1
            mov edi,pArr2
            mov ecx,lengthArr
            mov ebx,0
            compareElements:
                  mov edx,[esi + ebx * 4]
                  sub edx,[edi + ebx * 4]
                  cmp edx,maxDiff
                  jle incrementCount
                  jmp continueLoop
                  incrementCount:
                        inc eax
            continueLoop:
                inc ebx
        loop compareElements
        ret
    CountNearMatches ENDP
    END MAIN
```

Output:



```
The number of elements closer to difference  = 3
```