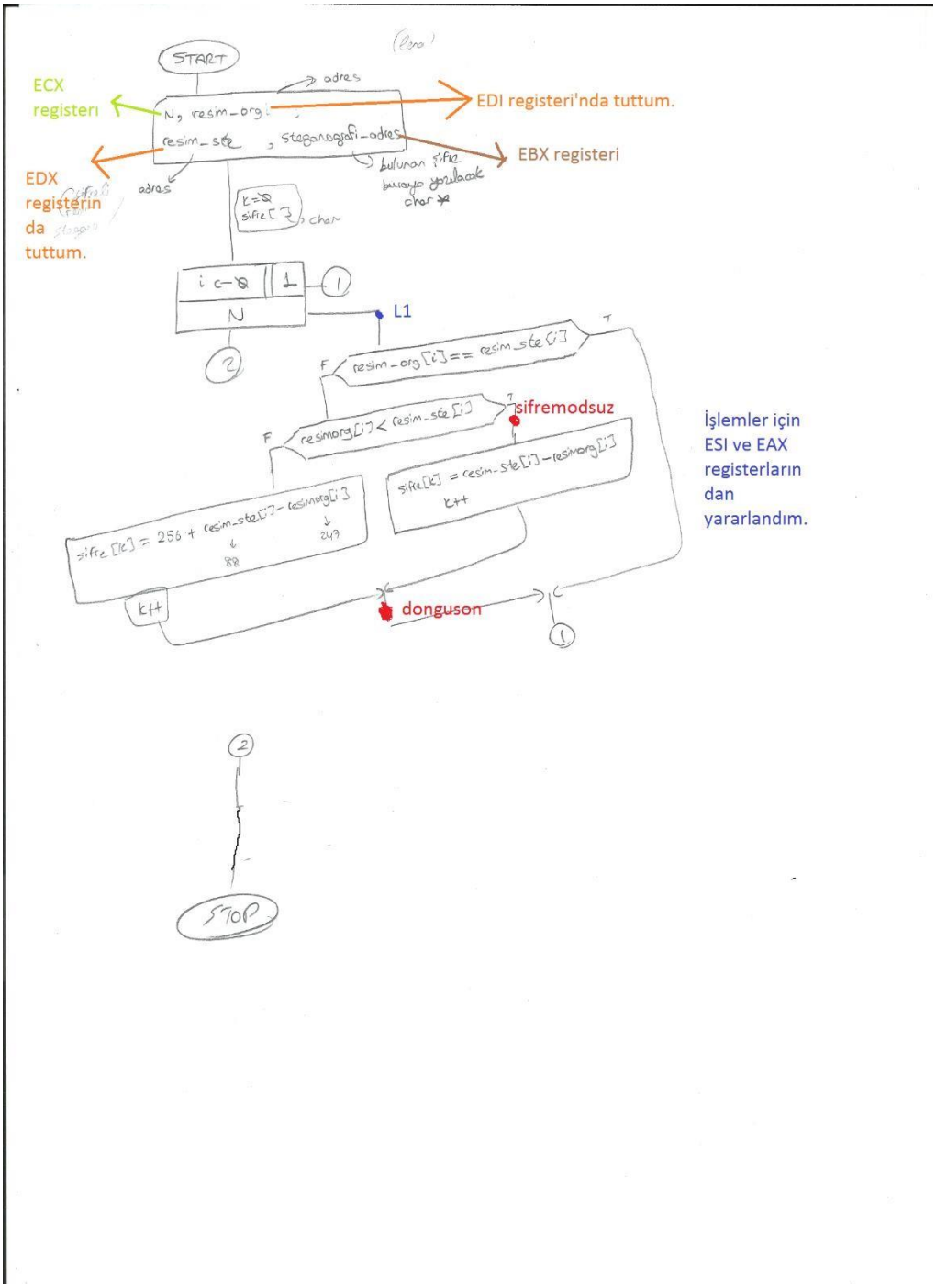
ASP ODEV_1 18011083

SORU 1:

İlk önce sorunun çözümü için bir algoritma yazdım. Daha açıklayıcı olması için resmini atıyorum.



Algoritmayı yazdıktan sonra assembly kodu şeklinde tutmak istediğim değerleri seçtiğim registerlara attım. Dallanma yapısını oluşturdum. Dizilere erişmek için adresleri kullanmak zorundaydım.

Adresleri registerlara attıktan sonra, döngü içinde i artınca bu (resim_org ve resim_ste) adresleri 2 ile artırdım. Çünkü diziler int tanımlıydı. Integer 2 byte olduğu için bunu yapmalıydım.

Öbür taraftan şifrenin tutulacağı dizinin adresi stegonagrafi_adres için ise bu artırma 1 ile olmalıydı çünkü bu dizi de char ile tanımlıydı. (1 byte)

Adreslere erişimi WORD/BYTE PTR [adres] , atanacak deger şeklinde yaptım. Kodların

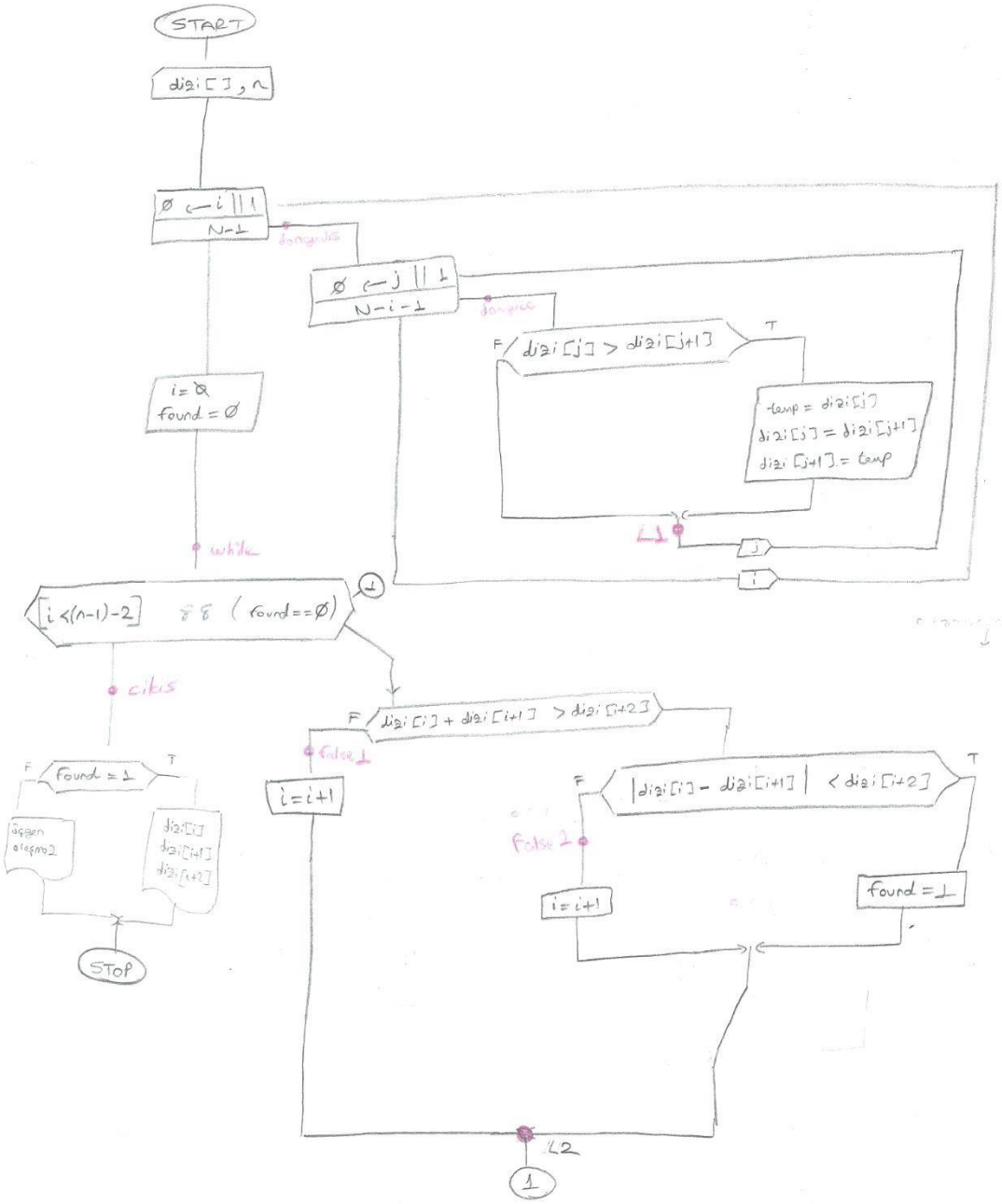
yanında detaylı açıklarsam;

```
MOV ECX, n                          //çevrim sayımı CX register'ına attım. N kere dönecek
MOV EDI, resim_org                  //resim_org adresine EDI reg ile erişeceğim
MOV EDX,resim_ste                   //resim_ste adresine EDX reg ile erişeceğim
MOV EBX, steganografi_adres         //şifrenin tutulacağı adrese EBX ile erişeceğim.

L1:  MOV AX, WORD PTR[EDI]
     CMP AX, WORD PTR[EDX]          //for döngüsünün içindeyim, iki dizi eşitse işlem
                                    // yapmıyorum atlıyorum
     JZ donguson

     CMP AX, WORD PTR[EDX]          //resim_org>resim_ste ise  255'I taşma olasılığı olmadığı
                                    // için bu case'i sifremodsuz label'inde uyguladım
     JL sifremodsuz

     MOV ESI, 0                     // şu anda resim_ste>resim_org ise olan case'i
                                    //uyguluyorum. Bu durumda ste dizisindeki değer 256'ya
                                    //göre modu alındığı için olması gerekenden 256 eksiktir.
                                    //Bu yüzden 256 ekliyorum. Sonra ise bu toplam ile
                                    //resim_ste[i]-resim_org[i] işleminin sonucunu
```

```
                                        //topluyorum. Çıkan sonuç şifrenin ASCII'si idir.
        SUB SI, WORD PTR[EDI]
        ADD ESI, 256
        ADD SI, WORD PTR[EDX]          // burada SI kullandım çünkü resim dizisi 16 bit
        MOV AX, SI                     //MOV komutunda her iki taraf aynı tip olmalı
        MOV BYTE PTR[EBX], AL          //Burada AL yi kullandım çünkü şifre dizisi 8 bit.
        INC EBX                        //taşma olma ihtimali olmadığı için yapabiliyorum bunu.
JMP donguson

sifremodsuz: MOV ESI,0                 // bu case resim_ste>resim_org yani şifrelenmiş
                                       //pikselde 256'ya göre mod alınmamış. Direct olarak
                                       //resim_ste-resim_org yaparak şifrenin ASCII'si buluyorum.
        ADD SI, WORD PTR [EDX]
        SUB SI,WORD PTR [EDI]
        MOV AX,SI
        MOV BYTE PTR [EBX],AL
        INC EBX
        JMP donguson

donguson: ADD EDI,2            //döngünün son işlemi olarak indis artırımı yapıyorum. Resim_ste ve
//resim_org için. Stegano_adress için artırmayı case içinde INC BX
//olarak zaten yapmıştım
        ADD EDX,2
        LOOP L1

//Buradan gerisi stringin yanına – 18011083 ifadesini ekliyor.( ASCII karşılıkları ile)
```

```
MOV BYTE PTR[EBX], 32
INC EBX
MOV BYTE PTR [EBX],45
INC EBX
MOV BYTE PTR[EBX], 32
INC EBX
MOV BYTE PTR[EBX], 49
INC EBX
MOV BYTE PTR[EBX], 56
INC EBX
MOV BYTE PTR[EBX], 48
INC EBX
MOV BYTE PTR[EBX], 49
INC EBX
MOV BYTE PTR[EBX], 49
INC EBX
MOV BYTE PTR[EBX], 48
INC EBX
MOV BYTE PTR[EBX], 56
INC EBX
MOV BYTE PTR[EBX], 51
INC EBX
MOV BYTE PTR [EBX],0
```

SORU 2: İlk önce algoritmamı yazdım. Verilen diziyi küçükten büyüğe doğru sıralayıp, o şekilde üçgen kontrolü yaptım. Algoritmayı atıyorum.


START
dizi[ ] , n
0 <- i || 1
N-1
dongucu
0 <- j || 1
N-i-1
dongucu
F
dizi[j] > dizi[j+1]
T
temp = dizi[j]
dizi[j] = dizi[j+1]
dizi[j+1] = temp
L1
j
i
i = 0
found = 0
while
[i<(n-1)-2] && (found==0)
1
cikis
F
found = 1
T
uygun olasma2
dizi[i]
dizi[i+1]
dizi[i+2]
STop
F
dizi[i] + dizi[i+1] > dizi[i+2]
false 1
i=i+1
F
|dizi[i] - dizi[i+1]| < dizi[i+2]
T
false 2
i = i+1
found = 1
L2
1

Programımı Assembly dilinde yazarken ilk önce data segmenti ve stack segmenti tanımladım. Ders kitabındaki ekrana veri yazdırma ve ekrandan veri alma fonksiyonlarını kullanabilmek için ANA (UCGEN) fonksiyonun dışında PUTN PUTC gibi fonksiyonları kod segmentin içine dahil ettim.

Kullanıcıya vermek istediğim mesajları ve kullandığım değişkenleri, dizimi data segmentte tanımladım.

Kodumda ilk önce kullanıcıdan dizinin eleman sayısını CALL GETN yordamıyla AX yazmacına aldım.

Alınan bu değer 2<=AX<100 aralığını aşıyorsa hata mesajı verecek şekilde ayarladım.

Girilen değer uygunsa bunu n değişkenimin içine aldım.

SI yazmacına dizinin başlangıç adresini atadım.

CX yazmacına n değerini atadım. (n kere döneceğiz.) Bu döngüde şimdi dizi elemanlarını alacağım.

Dizialma: label'I ile başladığım kod parçasında girilen değeri AX yazmacına alıp 0 ile 1000 arasında olup olmadığını control ettim. Uygun değilse hata verip elemanı yeniden isteyecek şekilde yazdım.

Girilen eleman uygunsa bunu diziye attım. SI yazmacını +2 yaptım ve LOOP ile bu döngüyü tekrarladım.

Artık dizi elemanlarını almış oldum. Şimdi en küçük çevreli üçgen kenarlarını bulabilmek için girilen bu diziyi küçükten büyüğe sıralamalıyım.

Bubble sort kullandım.

Dongudis ve donguic loopları ile diziyi sıraladım.

Dongudis n-1 kere, dongu ic ise n-i-1 kere döneceği için CX yazmacını buna göre ayarladım.

Dizi[i] 'nin adresini DI

Dizi[i+1]'nin adresini BX ile tuttum ve bu şekilde işlem yaptım.

Artık dizimiz sıralı, şimdi bu dizideki elemanları alıp üçgen kontrolü yapacağım.

While döngüsü açtım. Found değişkeninin değeri 1 olana kadar ve indis degişkeni dizinin sonuna taşmayana kadar bu döngü devam edecek.

Döngü içinde algoritmadaki gibi ilk üç elemanı alıp dizi[i]+dizi[i+1]>dizi[i+2]>|dizi[i]-dizi[i+1]|

Işlemi doğru mu kontrolü yaptım. Doğruysa found 1 oluyor ve döngüden çıkıp bu 3 kenarı yazdırıyoruz.

Eğer doğru değilse i değerini artırıp bir sonraki elemanlar için bunu deniyor.Bulursa döngüden çıkıyor.

Eğer hiçbiri uygun değilse, i+2 indisi n değerini aştığı an döngüden çıkılıyor ve üçgen oluşturulamadı error veriliyor. i+2 kontrol edilen son eleman olmalı. O yüzden i<n değil de i+2<n şeklinde şart koydum.

Eğer i<n aşıldığında döngüden çık deseydim farklı bir bellek alanına taşardım.

Programda diziye adresleri ile eriştiğim için i+2<n kontrolünü SI yazmacı ile yapamazdım. Bunun yerine indis adlı değişkeni kontrol amaçlı kullandım.

EK BİLGİ; döngü içindeyken

Dizi[i]'nin adresi SI

Dizi[i+1]'nin adresi BX

Dizi[i+2]'nin adresi DI yazmacında tuttum ve öyle işlem yaptım.

Döngüden çıkınca found ile ilgili kontrolümü yapıp gerekli mesajları yazdırıyorum.

Son olarak UCGEN proc fonksiyonumu ENDP ile bitiriyorum.

Buradan sonra kullandığım yordamlar var. PUT_STR gibi.

Bu yordamlar da bitince kod segmenti kapatıp programı bitirdim.

```
mystack SEGMENT PARA STACK 'yigin'
        DW 32 DUP(?)
mystack ENDS

datas SEGMENT PARA 'veri'

   CR     EQU 13            //Metin yazdırılırken yeni satırda ve sola doğru yazdırılsın.
   LF     EQU 10
   MSG1   DB  CR,LF, 'Ucgen dizisinin eleman sayisini giriniz:' ,0
   MSG2   DB  CR, LF,'Diziyi girin :',0
   MSG3   DB  CR,LF, 'Dizi elemani alindi.',0
   MSG4   DB  CR,LF, 'Dizi alimi basarili. Ucgen diziniz tamam.',0

   EL1    DB CR,LF, '1.kenar:',0
   EL2    DB CR,LF, '2.kenar:',0
   EL3    DB CR,LF, '3.kenar:' ,0

   HATA1  DB CR,LF, 'ELEMAN SAYISI GECERSIZ, 0-100 ARASINDA TAM SAYI GIRINIZ !' ,0
   HATA2  DB CR,LF, 'DIZIYE GECERSIZ SAYI GIRDINIZ, 0-1000 ARASINDA TAM SAYI GIRINIZ !'
,0
```

```
   HATA5  DB CR,LF, 'ERROR!' ,0

   SONUC1 DB CR,LF, 'Ucgeninizin kenarlari sunlardir:',0
   SONUC2 DB CR,LF, 'Bu verilerle ucgen olusturulamadi.',0
//İLGİLİ MESAJLARIN TANIMI.
//KULLANACAĞIM DEĞİŞKENLERİN TANIMI
   indis  DB 0
    n     DW ?
   dizi   DW 100 DUP(?)
   temp   DW ?
   found  DB 0

datas ENDS

//KOD SEGMENT BAŞLIYOR.
codesg SEGMENT PARA 'kod'
       ASSUME CS:codesg,DS:datas,SS:mystack

       UCGEN PROC FAR
        //exe TİPİ PROGRAM İÇİN BAŞLANGIÇ İŞLEMİ ( DATA SEGMENTE ERİŞMEK İÇİN)
         PUSH DS
         XOR AX,AX
         PUSH AX
         MOV AX,datas
         MOV DS,AX

  //BURADA DİZİNİN BOYUTUNU KULLANICIDAN ALIYORUM. EĞER GİRİLEN DEĞER YANLIŞSA SORUN1
LABEL'İ NE ZIPLIYORUM VE HATA MESAJI YOLLUYORUM. BURADAN TEKRAR JMP KOMUTU ILE
KULLANICIDAN TEKRAR DIZI BOYUTU GIRMESINI ISTEMEK ICIN diziboyut: LABEL'İNE GEÇİYORUM.
 diziboyut:
         MOV AX,OFFSET MSG1      //MSG1 mesajını yazdırmak için ilk önce offsetini ax'e
            aldım. Çünkü PUT_STR yordamı AX registerinda offseti olan mesajı yazdırıyor.

         CALL PUT_STR   //mesaj ekrana yazdırılıyor.
         CALL GETN      //KULLANICININ GIRDIGI SAYIYI ALAN FONKSIYON, ALINAN DEGER AX
                          REGISTER'INA YAZILIR.

         CMP AL,2         //ALINAN DEĞER 2<N<=100 ARASINDA OLMALI. ÇÜNKÜ UCGEN İÇİN EN AZ
ÜÇ KENAR BİLGİSİ LAZIM. 100'Ü AŞMAMA ŞARTI İSE PROGRAMIMIZI YAZARKEN İSTENEN KOŞUL.

         JLE sorun1
         CMP AL,100
         JA sorun1
         JMP devam1  //KONTROLLERDE SORUN YOK ISE PROGRAM DEVAM EDIYOR.


   sorun1:  // GİRİLEN DİZİ BOYUTU YANLIŞSA HATA MESAJI HATA1 YAZDIRILIYOR. SONRA
diziboyut: LABEL'İNE ZIPLIYORUM.
         MOV AX, OFFSET HATA1
         CALL PUT_STR
         JMP diziboyut



   devam1:    //PROGRAM DEVAM EDİYOR. ALDIĞIM DİZİ BOYUTUNU n DEĞİŞKENİNE ATIYORUM.
         MOV n,AX
```

```
         LEA SI,dizi  //DATA SEGMENTTE TANIMLADIĞIM DİZİ'YE (ÜÇGEN KENARLARINI TUTACAK
OLAN DİZİ) ADRESI ILE ERİŞECEĞİM. ADRESİNİ SI YA ALDIM.


         MOV CX,n   // DİZİ ELEMANLARINI ALMAK İÇİN KURDUĞUM DÖNGÜDE N KERE DÖNECEĞİM.


         MOV AX, OFFSET MSG2  // MSG2 yi yazdırıyorum. Dizi elemanlarını gir diyorum.
         CALL PUT_STR

dizialma:
        CALL GETN      //GIRILEN SAYI AX'e alındı.

             CMP AX,0          //GIRILEN SAYI 0<=X<=1000 koşullarını sağlamalıdır.
        JL sorun2              //koşul sağlanmıyorsa sorun2 label'ina gidilir.
        CMP AX,1000
        JA sorun2

             JMP devam2      //SAYILAR UYGUNSA AKIŞ DEVAM EDER.

sorun2: MOV AX, OFFSET HATA2   //HATA2 mesajını yazdır, dizialma label'ine geri don.
        CALL PUT_STR            //kullanıcıdan tekrar değer al.

             JMP dizialma

devam2: MOV WORD PTR [SI],AX      //girilen değeri diziye alıyorum.
        MOV AX, OFFSET MSG3       //eleman alındı mesajını yazdırıyorum.
        CALL PUT_STR

         ADD SI,2                 //sonraki elemana geçmek için sı 2 artırılır.(WORD
                                                                    tanımlı)


LOOP dizialma               //DONGU YAP.

//ARTIK DIZI ELEMANLARI ALINDI. ILGILI MESAJI YAZDIRIYORUM.

              MOV AX,OFFSET MSG4
              CALL PUT_STR

 //BUBBLE SORT ICIN FOR DONGUSUNU HAZIRLIYORUM. DIŞTAKİ DÖNGÜ N-1 KERE DÖNECEK.
//İÇTEKİ DÖNGÜ İSE N-1-İ KADAR DÖNECEK.
             MOV CX,n
              DEC CX
              MOV SI,0  //BURADA SI'YI İ (İNDİS) OLARAK KULLANDIM.


 dongudis:

              PUSH CX  //İÇTEKİ DÖNGÜ İÇİN CX KULLANILACAK, DEĞERİ SAKLAMALIYIM.
              PUSH n      //N DEĞİŞKENİNİ SAKLA.

              SUB n,SI   //N DEĞİŞKENİNİN DEĞERİ ARTIK N-İ OLDU.

              MOV CX,n
              DEC CX    // CX ARTIK N-İ-1 SAYISINI TUTUYOR.
```

```
        POP n      //N İLE İŞİM BİTTİ, ARTIK YIĞINDAN ALABİLİRİM. ESKİ DEĞERİ GELSİN.


              LEA DI,dizi      //DIZIYE DI İLE ERİŞECEĞİM.

                 donguic:
                                  //DI register'I dizinin dizi[i] adresinin değerini
tutarken,
//BX register'I dizinin dizi[i+1] değerinin adresini tutuyor.
//BÖYLECE KIYASLAMALARI BASİTÇE YAPABİLİRİM.


                                                MOV BX,DI
                                               ADD BX,2
                                               MOV AX,WORD PTR[DI]
                                               CMP AX,WORD PTR[BX]
//CMP KOMUTU İÇİN İKİ TARAF DA RAM OLAMAYACAĞI İÇİN BİR ÖGEYİ AX'E ATMAK ZORUNDAYIM.
//EĞER DİZİ[İ]<=DİZİ[İ+1] İSE L1 LABEL'İNE GİDİYORUM.
                                               JLE L1

                                            // EĞER dizi[i]>dizi[i+1] ise bu ikisinin
                                    yerini değiştiriyorum. (bubble sort böyle çalışıyor.)

                                               MOV temp,AX
//AX şuan dizi[i]'yi tutuyor.bunu temp değişkenine aldım.
//alttaki iki satırda ise dizi[i]=dizi[i+1] işlemini yaptım.

                                               MOV AX,WORD PTR[BX]
                                               MOV WORD PTR[DI],AX
//burada ise dizi[i+1]=temp; işlemini yaptım.
                                               MOV AX,temp
                                               MOV WORD PTR[BX],AX

                        L1: //artık işlemler tamamlandı, bir sonraki dönüm için dizinin
sonraki elemanına geçiyorum. +2 yapıyorum çünkü word tanımlı.
                                          ADD DI,2

                    LOOP donguic  //dongü ic 'I kuran komut. N-1-i kere dönecek.

           POP CX  //DONGUDIS İÇİN SAKLADIĞIM CX DEĞERİNİ ÇAĞIRIYORUM. DIŞ DÖNGÜ N-1 KERE
                                                                                DÖNECEK.
                 INC SI //sı REGİSTERİNİ İ OLARAK KULLANDIM. O YUZDEN 1 ARTIRIYORUM.
LOOP dongudis
//DONGUYU KURAN KOMUT.

//ARTIK DIZIMIZ KUCUKTEN BUYUGE DOGRU SIRALI. GERİYE YAPMAMIZ GEREKEN ELEMANLARI UCGEN
KONTROLUNE TABI TUTMAK.

                 LEA SI,dizi      //DIZIYE BU SEFER SI YAZMACIYLA ERİŞECEĞİM.

                          XOR CX,CX


           while:

                  XOR AX,AX
                  CMP found,1  //FOUND DEĞİŞKENİ 1 İSE WHİLE'DAN ÇIK.
```

```
                    JE cikis

                    MOV CL,indis //indis değerini cx'e alıp 2 artırıyorum.
                    ADD CX,2       //sonra bunu n ile kıyaslıyorum.
                CMP n,CX
 //BURADA YAPTIĞIM KONTROL ÇOK ÖNEMLİ. DİZİNİN İNDİSİ CX İLE TUTULUYOR. BURADA i değeri
(N-1)-2 değerini geçmemeli. Çünkü dizi[n-1] değeri dizimizin son değeri. Ben bu while
döngüsü içinde i=0 iken hem dizi[0], hem de dizi[1] ve dizi[2] ye erişiyorum. BU ŞU
DEMEK, SON GELDİĞİM İ DEĞERİ N-3'Ü AŞARSA dizimi aşar ve başka ram alanına girerim. Bunu
istemiyorum.
                JBE cikis


//while döngüsü içindeki işlemlerime başlıyorum. Dizi[i]'ye SI, dizi[i+1]'e BX, dizi
[i+2]' ye DI ile erişeceğim.

                    MOV BX,SI
                    ADD BX,2
                    MOV DI,SI
                    ADD DI,4

                    ADD AX,WORD PTR[SI]
                    ADD AX,WORD PTR[BX]
//dizi[i]+dizi[i+1] ile dizi[i+2] yi kıyaslıyorum. UCGEN KURALINI HATIRLAYALIM.
A+B>C>|A-B|, ben burada ilk kısmı control ediyorum. Zaten bubble sort kullanmamın amacı
da şuydu, dizinin ilk elemanı sonraki iki elemanla üçgen kuramıyorsa, bunlardan büyük
olan değerlerle de kuramaz. Yani tek bir seferde control yapıp program bitirebilirim.

                    CMP AX,WORD PTR[DI]
                    JBE false1
//eğer şart sağlanmıyorsa false1 label'ine git. Bu label ise dizinin sonraki elemanı için
artırımlar yapıyor.SI değeri ve indis değeri artırılıyor.
//ALTTAKİ KONTROL İSE A+B>C İŞLEMİ DOĞRUYSA YAPILAN , SIRADAKİ C>|A-B| KONTROLÜDÜR.
                  MOV AX,WORD PTR [SI]
                    SUB AX,WORD PTR[BX]
//EĞER A-B DEĞERİNİN SONUCU NEGATİİF GELİYORSA, MUTLAK DEĞERİNİ ALMALIYIZ.
       adim:  CMP AX,0
                JL negatif

//BURADA İSE |A-B| VE C DEĞERİNİ KIYASLIYORUM. Ax ŞUAN dizi[i]-dizi[i+1] (A-B) nin mutlak
değerini tutuyor.

                CMP AX,WORD PTR[DI]
                      JAE false2
               //EĞER |A-B|<C KOŞULU SAĞLANMIYORSA FALSE2 LABEL'İNA GİT.
//EĞER KOŞUL DOĞRUYSA EUREKA! En düşük çevreye sahip üçgeni bulduk.
                    MOV found,1  //SAYAÇ 1 olsun ki while'dan çıkalım.
                    JMP L2     //AKIŞA DEVAM ETMEK İÇİN.

negatif:      NEG AX
           JMP adim



     false2:  //koşullar sağlanmadıysa gerekli indis artırımı. Sıradaki elemana erişmek
                                          için.
            ADD SI,2
```

```
                MOV CL,indis
                INC CL
                MOV indis,CL
          JMP L2


   false1:  //aynı şekilde koşullar sağlanmadıysa sıradaki dizi elemanına erişmek için
yapılan işlemler.
          ADD SI,2

                MOV CL,indis
                INC CL
                MOV indis,CL
          JMP L2

//WHİLE SONUNA geldik. Jmp while ile donguyu kuruyorum.
                L2:
                   JMP while


//BURAYA GELDIYSEK WHILE'DAN ÇIKTIK DEMEK. WHILE'DAN ÇIKTIYSAK YA DİZİNİN SONUNA
GELMİŞİZDİR YA DA ÜCGENİ BULMUŞUZDUR. BUNUN KONTROLUNU YAPALIM.
//FOUND DEĞİŞKENİ BİR İSE BULUNDU: LABEL'İNE ATLIYORUM.
//FOUND DEĞİŞENİ 0 İSE BULUNAMADI: LABEL'İNE ATLIYORUM.

cikis:

       MOV AL,0
       CMP AL,found
          JE bulunamadi
          MOV AL,1
          CMP AL,found
          JE bulundu
          //EĞER OLDU DA BU IKISINE GIREMEDIYSEK PROGRAMDA CIDDI SIKINTI VAR DEMEKTİR.
//ÖNLEM OLARAK JMP bitir yazdım.
          JMP bitir


          bulundu: //burada eğer üçgen bulunduysa ilgili mesajları yazdırıyorum.
          MOV AX,OFFSET SONUC1  // üçgen bulundu bla bla
          CALL PUT_STR

          MOV AX,OFFSET EL1   //1. Kenarınız bla bla
          CALL PUT_STR
          MOV AX,WORD PTR[SI]  //burada 1. Kenarı AX'e atıp, PUTN ile AX'teki değeri
                                          ekrana yazdırıyorum.
          CALL PUTN


          MOV AX,OFFSET EL2   //2. Kenarınız bla bla

          CALL PUT_STR
          MOV AX,WORD PTR[BX]  //2. Kenarı yazdırıyorum aynı şekilde.
          CALL PUTN

          MOV AX,OFFSET EL3 //3. Kenarınız bla bla
```

```
          CALL PUT_STR
          MOV AX,WORD PTR[DI]  //3. Kenarı yazdırıyorum.
          CALL PUTN


  JMP bitir     //BITIR LABEL'ına git.


//BULUNAMADI LABEL'INA GELDİYSEK ÜCGEN KURULAMAMIS DEMEKTIR. UCGEN KURULAMADI MESAJINI
YAZDIRIYORUM.
  bulunamadi: MOV AX,OFFSET SONUC2
              CALL PUT_STR
//BURADA TEKRAR JMP BİTİR YAZMAYA GEREK YOK. HEMEN ALTI ZATEN ARADA KOD YOK.
bitir:


       //UCGEN FONKSIYONUMU BITIRIYORUM.



             RETF
       UCGEN ENDP

//UCGEN FONKSIYONUMU BITIRDIM, AMA KOD SEGMENTİ KAPATMADIM. BURAYA EKRANA VERI YAZDIRMAK
İÇİN KULLANDIĞIM YORDAMLARIN TANIMLARINI YAZDIM.


PUT_STR PROC NEAR  //EKRANA AX YAZMACINDA OFFSETİ VERİLEN, SONUNDA 0 OLAN DİZGEYİ
YAZDIRIR.

    PUSH BX
    MOV BX,AX
    MOV AL,BYTE PTR [BX]

 PUT_LOOP:
    CMP AL,0
    JE PUT_FIN
    CALL PUTC
    INC BX
    MOV AL,BYTE PTR [BX]
    JMP PUT_LOOP
 PUT_FIN:
    POP BX
    RET
PUT_STR ENDP


PUTC PROC NEAR  //AL YAZMACINDAKİ DEĞERİ EKRANA YAZDIRIR.

    PUSH AX
    PUSH DX
    MOV DL,AL
    MOV AH,2
    INT 21H
    POP DX
    POP AX
```

```
    RET
PUTC ENDP

GETC PROC NEAR  //KLAVYEDEN BASILAN DEĞERİ AL YAZMACINA ALIR VE EKRANDA GÖSTERİR.
                //SADECE AL'NİN DEĞERİ KAYBOLUR.
    MOV AH, 1H
    INT 21H
    RET
GETC ENDP

 GETN PROC NEAR  //KLAVYEDEN BASILAN SAYIYI OKUR. SONUÇ AX'TE OLUŞUR.

    PUSH BX
    PUSH CX

  GETN_START:

    XOR BX,BX
    XOR CX,CX
  NEW:
    CALL GETC
    CMP AL,CR
    JE FIN_READ
  CTRL_NUM:
    CMP AX,0
    JL error
    CMP AX,1000
    JA error

    SUB AL, '0'
    MOV BL,AL
    MOV AX,10

    PUSH DX
    MUL CX
    POP DX

    MOV CX,AX
    ADD CX,BX
    JMP NEW
  ERROR:
    MOV AX,OFFSET HATA5
    CALL PUT_STR
    JMP GETN_START
  FIN_READ:
    MOV AX,CX

  FIN_GETN:

    POP CX
    POP BX
    RET
GETN ENDP

PUTN PROC NEAR  //AL'DE BULUNAN SAYIYI EKRANA YAZDIRIR.
    PUSH CX
    PUSH DX
```

```
    XOR DX,DX
    PUSH DX

    MOV CX,10
    CMP AX,0
    JGE CALC_DIGITS

    NEG AX
    PUSH AX
    MOV AL, '-'
    CALL PUTC
    POP AX
  CALC_DIGITS:
    DIV CX
    ADD DX,'0'
    PUSH DX
    XOR DX,DX
    CMP AX,0
    JNE CALC_DIGITS

  DISP_LOOP:

   POP AX
   CMP AX,0
   JE END_DISP_LOOP
   CALL PUTC
   JMP DISP_LOOP

 END_DISP_LOOP:

   POP DX
   POP CX
   RET
PUTN ENDP



codesg ENDS
END UCGEN  //CODE SEGMENTİ KAPATIP PROGRAMI BİTİR.
```