YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BLM3021 – ALGORİTMA ANALİZİ

3.ÖDEV 2.SORU RAPORU

Konu : Sorgulanan bir cümlede yanlış yazılmış kelimeler varsa bu kelimelerin yerine doğru kelimeler öneren bir sistem tasarlanması

Ad Soyad : Eren TUTUŞ

Okul No : 17011702

E-posta : l1117702@std.yildiz.edu.tr

Programın Çalışma Süreci

1-) void baslangicKeyDegeri ( ) :

Hash tablosunda ve hatalı kelime tablosunda key ve kelimeler yerleştirmeden önce her key değerini -1 değeri ile işaretledik. Böylelikle tablolara değerler ekledikten sonra boş kalan indislerin hangileri olduğunu o indisin key değerini -1 ile karşılaştırıp elde etmiş olacağız.

2-) void ekleHashTable (int key, char word [ ]) :

Hash tablosuna verileri eklerken bu fonksiyona bilgiler gönderiyoruz ve burada key değerinin uygun indise yerleşmesi için 2 tane hashten geçirmemiz gerekiyor çünkü bu ödevimizde double hashing mantığını kullanıyoruz. İlk olarak key değeri double hashingten geçtikten sonra eğer uygun indis -1 olarak işaretliyse oraya daha önce hiç key değeri gelmemiş yani çakışma yaşanmayacak ve direk o indise key değerini ve kelimeyi ekleyebiliriz. Fakat gelen key değerine göre belirlenen indiste daha önce key değeri oluşmuş ise i değerini arttırıp yeniden hashing işlemi yapıyoruz bunun nedeni aynı indise değer yazmayı önlemek yani çakışmayı önlemek. Bu aşamada i değeri sürekli artarak key değeri kendisine boş bir indis bulana dek bu işleme devam edilir. Burada önemli bir diğer husus ise i değeri artarken döngü içerisinde bir şart koymamız gerekiyor i < TABLE_SIZE çünkü i değeri tablo boyutunu aşmamalı. Key değeri kendisine uygun bir indis bulduktan sonra bu indise key değerini ve kelimeyi yazacağız fakat burada bir sorgu daha yapmamız gerekiyor, bu sorgu gelen kelime adı daha önce hash tablosunda aynı indis içerisinde eklenmiş mi. Eğer aynı kelime adı varsa bir daha aynı kelime adını yazmayacak.

3-) void dokumanOku ( ) :

Doküman okuma fonksiyonunda, dökümandan gelen kelimeleri okuyup kelimeler için horner metoduna göre key değeri oluşturup bunu 2.fonksiyon olan ekleHashTable fonksiyonuna yollayıp hash table üzerinde gerekli indise yerleştiriyoruz. Sözlükte kelimeler aralarında birer boşluk var bu nedenle sözlükten kelimeleri ayırt edebilmek için ‘ ‘ (boşluk karakteri) ve ‘ \n ‘ (satır sonu karakteri) karakterlerine göre sorgulama yapıyoruz.

4-) void kelimeAra (char word [ ]) :

Bu fonksiyon sayesinde aranan kelime hash tablosunda var olup olmadğının sorgusunu yapıyoruz. Fonksiyona gelen kelimenin key adresini bilirsek hash tablosunda bulmak bizim için daha kolay olacak bu nedenle kelimeyi yeniden horner metodundan geçirip key değerini buluyoruz ve bu key değerinide tekrar double hashing'den geçiriyoruz. Aranan kelime zaten hash tablosunda mevcut ise kullanıcıya bu kelimenin zaten tabloda olduğunu bildiren bir çıktı verecek. Ancak kelime yoksa o halde önerilen kelimeler varmı bunu inceleyeceğiz. Aranan kelimenin hash tablosunda olmadığı senaryoyu düşünecek olursak;

Öncelikle kullanıcının girdiği kelimeyi tablodaki tüm kelimelerle karşılaştırmamız gerektiği için tabloda key değeri -1 den farklı olan tüm key değerleri bize kelimeleri vereceği için bu sorgulamayı yapıyoruz daha sonra bu kelimeleri tek tek LevenshteinEditDistance (word, hash[i].word) şeklinde yolluyoruz buradan dönen distance değerine göre önce mesafesi 1 olanlar varsa onları yazdırıyoruz fakat mesafesi 1 olan kelime yoksa o zaman mesafesi 2 olan kelimeleri yazdırıyoruz. Mesafesi 2 den büyük olanları ise almıyoruz bunun için bize sorulan bonus sorusunda yapmamız gerekenleri aşağıda LevenshteinEditDistance ( ) fonksiyonunda daha detaylı açıklıyor olacağım.

5-) int dogruKelimeyiAra (char word [ ]) :

Bu fonksiyonda biz tabloda hash tablosunda olmayan bir kelimeyi girdiğimizde bize önerilen kelimeler arasından seçeceğimiz kelimenin gerçektende hash tablosunda var olup olmadğını sorguluyoruz. Parametre olarak gelen kelimeyi yine horner metodundan geçirip key değeri oluşturup bu key değerini hash tablosunda arıyoruz.

6-) int hornerMetodu (char word [ ]) :

Horner metodunda R = 31 aldım ve kelime için key değeri oluşturduğumuzda bu key değeri int veri tipini aşmakta bunun için unsigned long long int olarak tanımladım key değerini. Dökümanda kelimeler aralarında birer boşluk var. Bu nedenle dökümandan kelimeleri ayırt edebilmek için ' ' (boşluk karakteri) ve ' \n ' (satır sonu karakteri) karakterlerine göre sorgulama yapıyoruz.

Horner metodunda key = R^(L-1) * (str[0] – ‘A’ + 1) + ……. R^0 * (str[L-1] – ‘A’ +1) formülünü kullanacağız burada R = 31 , L = Kelime uzunluğu, str = Kelime.

7-) int hataliTablodaArama (int key, char kelime [ ]) :

Bunu bir senaryo üzerinden canlandıracak olursak ;

Daha önce kullanıcının girdiği bir kelime için önerilen kelimeler listelenmiş olsun ve bu önerilen kelimeler içerisinden kullanıcının herhangi bir kelimeyi seçtiğini varsayalım. O halde bu iki kelime hatalı tablosuna eklenmiş olacak. O zaman tekrar yapılan bir aramada kullanıcının daha önce girmiş olduğu kelime için artık seçmiş olduğu önerilen kelime kullanıcıya tekrar getirebilmek için hatalı tabloda bu kelimeyi arama işlemi yapıyoruz.

8-) int ekleHataliTablo (int key, char hataliKelime [ ] ,char onerilenKelime [ ]) :

Daha önce tabloya ekleme işlemini hash tablosu ekleme fonksiyonunda yapmıştık burada da aynı işlemleri yaptık bundan farklı olarak önerilen kelimeyide hatalı kelime ile aynı key değerinde tuttuk.

9-) int enKucukBul [int x, int y, int z]) :

Edit Distance için minimum değer bulurken insert, delete ve change işlemleri arasından en küçüğü bulan fonksiyon.

10-) int LevenshteinEditDistance (char ilkKelime [ ] , char ikinciKelime [ ]) :

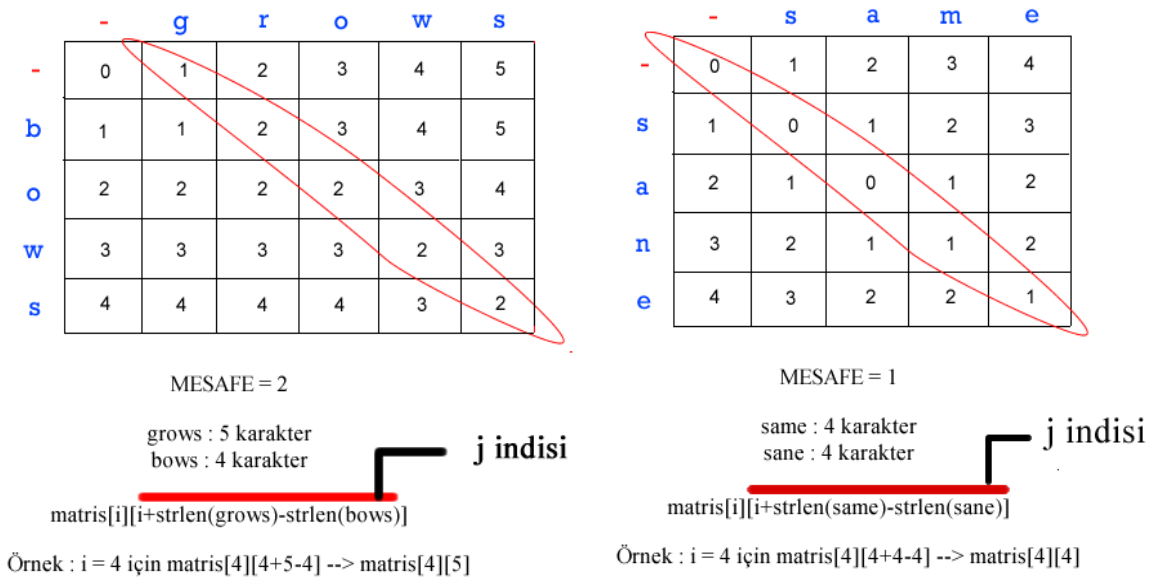
Bu fonksiyonda iki kelime arasındaki mesafeyi bulacağız. Öncelikle iki kelimenin karakter sayısının bir fazlası kadar satır ve sütun sayısı olacak şekilde matris oluşturuyoruz. Bunun sebebi en baştaki boşluk karakteri. Daha sonra her kelimenin boşluğa olan uzaklıklarını initialize ediyoruz yani 5 karakterli kelimenin boşluğa uzaklığı 5. Daha sonra matrisle ilgili indislere mesafeleri eklemek için iç içe iki döngü içinde geziniyoruz. Harf harf sorgu yaptığımız için harfleri kontrol ediyoruz eğer harfler eşit ise copy işlemi yapıyoruz yani

cost(+1) yapmıyoruz. O zaman aynı harf gelirse sol çapraz üstündeki mesafeyi alabiliriz. Fakat harf aynı değil ise o zaman insert, delete ve change işlemlerini gerçekleştirip bu değerleri 9.fonksiyonumuz olan enKucukBul fonksiyonuna yolluyoruz ve dönen değeri matris[i][j] ye ekliyoruz.

BONUS BÖLÜMÜ

enKucukBul fonksiyonundan dönen değer 2'den büyük ve mevcut indis yani iki kelimenin farkının i ile toplamı j'ye eşit ise o zaman matris[strlen(ilkKelime)][strlen(ikinciKelime)] indisine diyagonal olan indislerde 2'den büyük bir mesafe değeri yerleşti. Diyagonelde bulunan tüm mesafeler normalde 2'den büyük olmaması gerekir eğer 2 den büyük ise o zaman fonksiyon orada sonlanacak.

Resimle anlatacak olursak ;



Eğer bonus bölümündeki koşullar gerçekleşmez ise o zaman diyagonalde bulunan mesafeler en son dönecek mesafe değerinden küçüktür ve bu dönen en son mesafe değeride 2'den küçüktür.

10-) int main ( ) :

baslangicKeyDegeri ( ) fonksiyonunu çağırarak öncelikle hash tablosunda tüm key değerini -1 olarak işaretliyoruz. Sonra dokumanOku ( ) fonksiyonunu çağırarak sözlükteki kelimeleri hash tablosuna aktarıyoruz. Daha sonra kullanıcıdan bir cümle girilmesi isteniyor ve bu cümlenin kelimelerini aradaki boşluk karakterini sorguda kullanarak ayırıp tek tek işleme sokuyoruz. İlk olarak aranan kelime daha önce önerilen kelime ile birlikte hatalı kelime tablosuna eklenmiş mi bunu sorguluyoruz. Eğer eklenmemiş ise o zaman bu kelime hash tablosunda mevcut mu , yok ise bu kelime için önerilen bir kelime var mı ya da bu kelime hem hash tablosunda mevcut değil ayrıca önerilen bir kelimede bulunamadığını kullanıcıya sunuyoruz. Burada sorguladığımız diğer bir detay ise önerilen kelimeler içinden kullanıcı bir kelime seçtiğinde gerçekten de hash tablosundaki bir kelimeyi mi seçti bu durumu da sorguluyoruz.

C KODU

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define TABLE_SIZE 999983  // 1000000'den küçük en büyük asal sayıyı tablo boyutu olarak aldık.

typedef struct link{
    unsigned long long int key; // Key değeri int'in alan boyunu geçtiği için unsigned long long int tanımladım.
    char word[50];  // Kelimeleri tuttuğum yer.
}hashTable;

typedef struct link2{
    unsigned long long int key;
    char hataliKelime[50];   // Hatalı kelime ve Önerilen doğru kelimeyi tuttuğum yer
    char onerilenKelime[50]; // Önerilen kelimeyi hatalı kelimenin yanına eklersek bir sonraki hatalı kelime arayışımızda önceden hata aldığımız değer gelirse yanındaki önerilen kelimeyi direk çıktı olarak verecek.
}hataliKelimeTablosu;

hashTable hash[TABLE_SIZE];     // Hash Tablosu
hataliKelimeTablosu hash2[TABLE_SIZE];  // Hatalı Kelime Tablosu

void baslangicKeyDegeri(){   // Tabloda key değerlerini ilk başta -1 olarak işaretleme yapacağım fonksiyon.
    int i;
```

```
    for(i=0;i<TABLE_SIZE;i++){
        hash[i].key = -1;  //  Böylelikle içinde değer olmayan satırlar -1
olarak gösterilmiş oldu.
        hash2[i].key = -1;
    }
}

void ekleHashTable(int key,char word[]){  // Tabloya yerleştirilecek olan
bilgilerin işlendiği fonksiyon.
    int i = 0;
    int ilkHash = key % TABLE_SIZE;  // Double hashing'de iki kez hash'den
geçirmemiz gerekiyor key değerini
    int ikinciHash = 1 + (key % (TABLE_SIZE - 1) ); // İkinci hash'den
geçirdik.
    int calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE;  //  i =
0 değerine göre key için gerekli indis bulundu.

    if (hash[calculatedHash].key == -1){  // Eğer gelen key değeri için
belirlenen indiste -1 mevcutsa o zaman ilk defa o indise key atanacaktır.
        hash[calculatedHash].key = key;
        strcpy(hash[calculatedHash].word,word);
    }
    else{// else durumunda ise hash tablosunda belirlenen indiste -1 yok
yani bu durum bize daha önce o indise bir key değeri yerleşmiş olduğunu
belirtiyor.
        while(i<TABLE_SIZE){ // i değeri tablo boyutunu açmayacak şekilde;
          //  gelen key değeri -1'den farklı mı , gelen key değeri için
daha önce aynı indiste bir key yerleşmiş mi  kontrollerini yaparız.
            if(strcmp(hash[calculatedHash].word, word) != 0){
                calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE;
// böylelikle kendisine boş indis bulana dek bu döngü i sayacımız sayesinde
devam eder.
                if (hash[calculatedHash].key == -1 )
                {
                    hash[calculatedHash].key = key;  // tabloya
yerleştiriliyor.
                    strcpy(hash[calculatedHash].word,word);
                }
            }
            i++;
        }
    }
}

void dokumanOku(){  // Dökümandan gelen kelimeleri okuyup kelimeler için
horner metoduna göre key değeri oluşturup bunu ekleHashTable fonksiyonuna
yollayıp hashtable üzerinde gerekli indise yerleştiriyoruz.
    FILE *fp;
    if((fp = fopen("smallDictionary.txt","r"))== NULL){   // Dosya okuma
kontrolü
        printf("Dosya acma hatasi");
    }
    int i=0,j;
    unsigned long long int key=0;  // key değeri int değerini aşacağı için
unsigned long long int yaptım.
    char temp[50];   // Dökümandaki tüm karakterleri kontrol olarak
kullanmak için alacağım geçici dizi
    char temp2[50];  // Dökümandaki tüm kelimeleri alacağım geçici dizi
    while(!feof(fp)){
        temp[i] = fgetc(fp);
```

```
        if(temp[i] == ' ' || temp[i] == '\n'){   // Dökümanda kelimeler
arasında boşluk (' ') olacağı için kontrol yapıyorum her boşlukta kelimeyi
alıp hash tablosuna aktarıyorum.
            key = hornerMetodu(temp2);
            key = abs(key);
            if (key != 0){
                ekleHashTable(key,temp2);  // Gelen key değeri,kelime ve
dosyaAdina göre Hash tablosuna eklenmek üzere ekleHashTable fonksiyonuna
yollanıyor.
            }

            for(j=0;j<50;j++){  // Gecici temp2 degiskeninin içini
boşaltıyoruz.
                temp2[j] = '\0';
            }
            i=0;
        }
        else{
            temp2[i] = temp[i];  // temp ile son işlemde her zaman özel
karakter geleceği için kelimeyi temp2'de tutuyorum.
            i++;
        }
    }
    fclose(fp);
}

int kelimeAra(char word[]){  // Bu fonksiyon sayesinde aranan kelime hash
tablosunda var mı ? eğer var ise hangi dökümanlarda yer aldığını
kullanıcıya çıktı olarak verecek.
    int i=0,j;
    unsigned long long int key;  // key değeri int değerini aşacağı için
unsigned long long int yaptım.
    key = hornerMetodu(word);
    int ilkHash = key % TABLE_SIZE;  // Double hashing'de iki kere hash'den
geçirip indis değeri üreteceğimiz için
    int ikinciHash = 1 + (key % (TABLE_SIZE - 1) ); // İkinci hash işlemi
    int calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE; // Ve
indis değeri oluştu.
    int mesafeBirVarmi=0;
    int mesafeIkiVarmi=0;
    if(hash[calculatedHash].key == key && strcmp(hash[calculatedHash].word,
word) != 1){   // İlgili indiste bulunan key değeri ile kullanıcının
göndermiş olduğu kelimenin key değeri birbirine eşit mi ?
        printf("\nKelime hash tablosunda zaten mevcut :
%s",hash[calculatedHash].word);
        return 1;
    }
    else{
        int distance; // LevenshteinEditDistance fonksiyonundan dönecek
olan Distance(Mesafe) değeri.
        for(i=0;i<TABLE_SIZE;i++){
            if(hash[i].key != -1){
                distance = LevenshteinEditDistance(word,hash[i].word); //
Tablodaki tüm kelimeler, klavyeden girilen kelime ile mesafesi bulunmak
üzere LevenshteinEditDistance fonksiyonuna gönderilir.
                if(distance == 1){  // Mesafesi 1 olanlar
                    printf("\n'%s' icin bu kelimeyi mi aradiniz (Mesafe %d)
: %s",word,distance,hash[i].word);
                    mesafeBirVarmi = 1; // Mesafe 1 olan kelimeler var ise
ekranda onlar gösterilmeli bu yüzden kontrol işareti koyuyoruz.
                }
```

```
            }
        }
        if(mesafeBirVarmi != 1){ // Eğer mesafeBirVarmi != 1 ise o zaman
mesafe 1 olan kelime hiç yoktur.
            for(i=0;i<TABLE_SIZE;i++){
                if(hash[i].key != -1){
                    distance = LevenshteinEditDistance(word,hash[i].word);
// Yine tablodaki tüm kelimeler ile klavyeden girilen kelimelerin mesafesi
hesaplanır.
                    if(distance == 2){  // Mesafesi 2 olanlar
                        printf("\n'%s' icin bu kelimeyi mi aradiniz (Mesafe
%d) : %s",word,distance,hash[i].word);
                        mesafeIkiVarmi = 1; // Mesafe 1 olmayıp, Mesafe 2
olan kelimeler var ise ekranda onlar gösterilmeli bu yüzden kontrol işareti
koyuyoruz.
                    }
                }
            }
        }
        if(mesafeBirVarmi == 0 && mesafeIkiVarmi == 0){ // Eğer Mesafe 1 ve
Mesafe 2 için kelime bulunamazsa o halde aranan kelime hem hash tablosunda
yok hem de mesafesi 2den büyüktür.
            return 2;
        }
    }

}

int dogruKelimeyiAra(char word[]){  // Bu fonksiyon sayesinde aranan kelime
hash tablosunda var mı ? eğer var ise hangi dökümanlarda yer aldığını
kullanıcıya çıktı olarak verecek.
    int i=0,j;
    unsigned long long int key;  // key değeri int değerini aşacağı için
unsigned long long int yaptım.
    key = hornerMetodu(word); // kelimeye ait key değeri bulmak için horner
metodundan geçirdik.
    key = abs(key);
    int ilkHash = key % TABLE_SIZE;  // Double hashing'de iki kere hash'den
geçirip indis değeri üreteceğimiz için
    int ikinciHash = 1 + (key % (TABLE_SIZE - 1) ); // İkinci hash işlemi
    int calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE; // Ve
indis değeri oluştu.
    if(hash[calculatedHash].key == key && strcmp(hash[calculatedHash].word,
word) != 1){   // İlgili indiste bulunan key değeri ile kullanıcının
göndermiş olduğu kelimenin key değeri birbirine eşit mi ?
        return 1; //  1 : Kelime var
    }
}

int hornerMetodu(char word[]){
    int i=0,j,k=0;
    int fark = 'a' - 'A';
    unsigned long long int key=0;  // key değeri int değerini aşacağı için
unsigned long long int yaptım.
    int length = strlen(word);
    int power = length-1; //  Horner metodunda R^(Length-1) alacağı için üs
değerini belirliyoruz ve döngüde 0 a kadar azalacak şekilde işleme
sokuyoruz.
    char gecici;
    while (word[k] != '\0') {
```

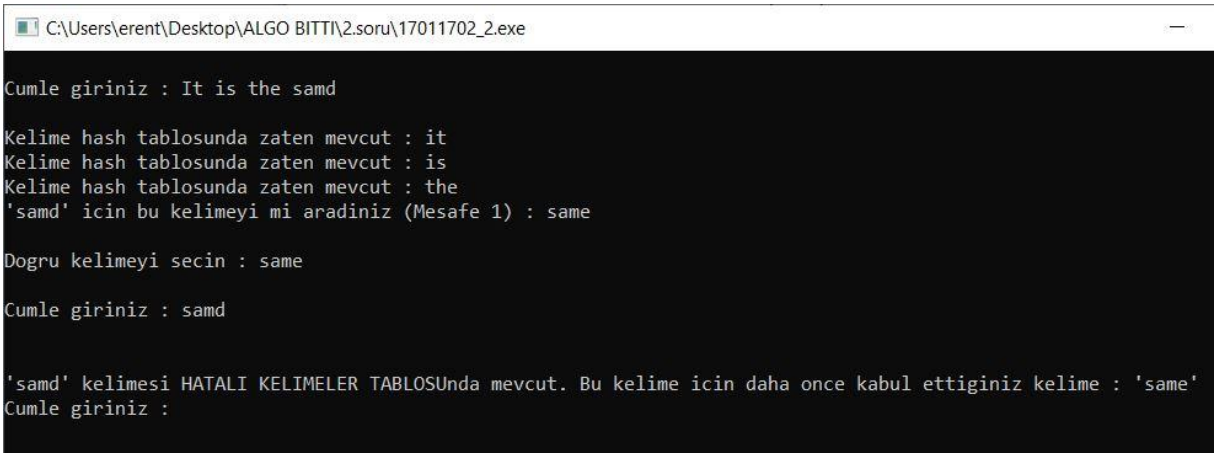
```
        if (word[k] >= 'A' && word[k] <= 'Z') { //harf büyük ise
küçültüyoruz..
            word[k] += fark;
        }
        k++;
    }
    for(j=0;j<length;j++){
        gecici = word[j];
        if (gecici >= 'a'){  // Araba ve araba aynı kelimeler gibi
düşüneceğiz bu nedenle kontrolü gerçekleştirdik.
            gecici = 'A' - 'a' + gecici;
        }
        key += abs((pow(31,power)*(gecici-'A'+1)));  // Horner metodunda
gelen kelimenin karakter sayısını kullanarak döngüye sokuyoruz işlem
bittiğinde kelime için key değeri oluşacaktır.
        power--;
    }
    key = abs(key);
    return key;  // key değerini döndürüyoruz.
}

int hataliTablodaArama(int key,char kelime[]){ // Hatalı tabloda arama
fonksiyonu
    int i=0;
    key = abs(key);
    int ilkHash = key % TABLE_SIZE;  // Double hashing'de iki kere hash'den
geçirip indis değeri üreteceğimiz için
    int ikinciHash = 1 + (key % (TABLE_SIZE - 1) ); // İkinci hash işlemi
    int calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE; // Ve
indis değeri oluştu.
    if(hash2[calculatedHash].key == key &&
strcmp(hash2[calculatedHash].hataliKelime, kelime) != 1){   // İlgili
indiste bulunan key değeri ile kullanıcının göndermiş olduğu kelimenin key
değeri birbirine eşit mi ?
        printf("\n\n'%s' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu
kelime icin daha once kabul ettiginiz kelime :
'%s'",hash2[calculatedHash].hataliKelime,hash2[calculatedHash].onerilenKeli
me);
        return 1; // Daha önce kelimenin hatalı kelime tablosunda var
olduğunu belirten kontrol değeri.
    }
    else{
        return 0;
    }
}

void ekleHataliTablo(int key,char hataliKelime[],char onerilenKelime[]){
    int i = 0;
    int ilkHash = key % TABLE_SIZE;  // Double hashing'de iki kez hash'den
geçirmemiz gerekiyor key değerini
    int ikinciHash = 1 + (key % (TABLE_SIZE - 1) ); // İkinci hash'den
geçirdik.
    int calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE;  //  i =
0 değerine göre key için gerekli indis bulundu.

    if (hash2[calculatedHash].key == -1){  // Eğer gelen key değeri için
belirlenen indiste -1 mevcutsa o zaman ilk defa o indise key atanacaktır.
        hash2[calculatedHash].key = key;
        strcpy(hash2[calculatedHash].hataliKelime,hataliKelime);
        strcpy(hash2[calculatedHash].onerilenKelime,onerilenKelime);
    }
```

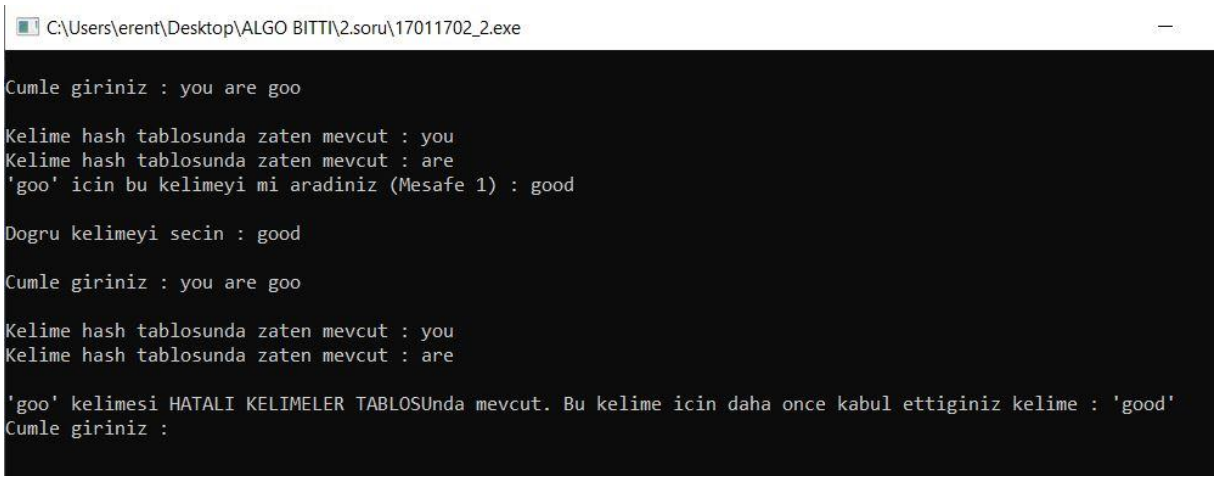
```
    else{// else durumunda ise hash tablosunda belirlenen indiste -1 yok
yani bu durum bize daha önce o indise bir key değeri yerleşmiş olduğunu
belirtiyor.
        while(i<TABLE_SIZE){ // i değeri tablo boyutunu açmayacak şekilde;
        //  gelen key değeri -1'den farklı mı , gelen key değeri için daha
önce aynı indiste bir key yerleşmiş mi  kontrollerini yaparız.
            if(strcmp(hash2[calculatedHash].hataliKelime, hataliKelime) !=
0){
                calculatedHash = (ilkHash + i * ikinciHash) % TABLE_SIZE;
// böylelikle kendisine boş indis bulana dek bu döngü i sayacımız sayesinde
devam eder.
                if (hash2[calculatedHash].key == -1 )
                {
                    hash2[calculatedHash].key = key;

strcpy(hash2[calculatedHash].hataliKelime,hataliKelime);

strcpy(hash2[calculatedHash].onerilenKelime,onerilenKelime);
                }
            }
            i++;
        }

    }
}

int enKucukBul(int x,int y, int z){ // Insert , Delete ve Change işlemleri
için en küçüğü bulan fonksiyon
    if(x<=y && x<=z){
        return x;
    }
    else if(y<=x && y<=z){
        return y;
    }
    else{
        return z;
    }
}


int LevenshteinEditDistance(char ilkKelime[],char ikinciKelime[]){  //
Mesafe bulduğumuz fonksiyon

    int i,j;
    int ilkKelimeLength = strlen(ilkKelime); // 1.kelime karakter sayısı
    int ikinciKelimeLength = strlen(ikinciKelime); // 2.kelime karakter
sayısı
    int insert,delete,change,copy;
    char temp,temp2;

    int matris[ilkKelimeLength+1][ikinciKelimeLength+1]; // kelimelerin 1
fazlasının olma sebebi en başta (0x0) boş karakter olarak almamız.

    for(i=0;i<=ikinciKelimeLength;i++){
        matris[0][i] = i;   // İlk Satırı initialize ettik boş karakterlere
olan uzaklıklarını kullanarak.
    }
    for(i=0;i<=ilkKelimeLength;i++){
        matris[i][0] = i;   // İlk Sütunu initialize ettik boş karakterlere
olan uzaklıklarını kullanarak.
    }
```

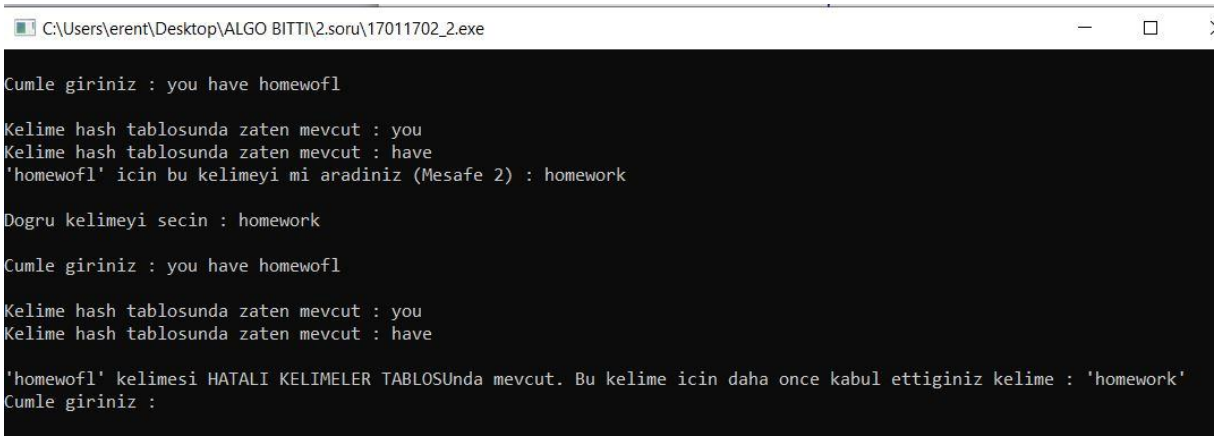
```
    for(i=1;i<=ilkKelimeLength;i++){  // Matriste ilgili indislere
Mesafe'leri eklemek için geziniyoruz.
        temp = ilkKelime[i-1];  // i=1'den başladığımız için i-1 yaptık.
        for(j=1;j<=ikinciKelimeLength;j++){
            temp2 = ikinciKelime[j-1]; // j=1'den başladığımız için j-1
yaptık.
            if(temp == temp2){
                copy = matris[i-1][j-1];  // Aynı harf gelirse sol çapraz
üstündeki Mesafe'yi alabiliriz. Copy işleminde aynı harf olduğu için COST
olmaz.
                matris[i][j] = copy;
            }
            else{
                insert = matris[i-1][j]+1; // Insert için Mesafe işlemi
                delete = matris[i][j-1]+1; // Delete için Mesafe işlemi
                change = matris[i-1][j-1]+1; // Change için Mesafe işlemi
                matris[i][j] = enKucukBul(insert,delete,change);  //
Insert, Delete ve Change'den dönen Mesafe'lerden en küçüğünü matris[i][j]
ye ekleyeceğiz.
                if(matris[i][j] > 2 && (( ilkKelimeLength-
ikinciKelimeLength + j) == i)){  // ****** BONUS ********  Alt satırlarda
açıkladım ama raporda daha detaylı açıkladım.
                    // enKucukBul fonksiyonundan dönen değer 2'den büyük VE
                    // mevcut indis yani iki kelimenin farkının j ile
toplamı i'ye eşit ise o zaman matris[ilkKelimeLength][ikinciKelimeLength]
indisine diyagonal olan indislerde 2'den büyük bir mesafe değeri yerleşti.
                    return -1;
                }
            }
            temp2 = '\0'; // geçici karakteri sıfırladık.
        }
        temp = '\0'; // geçici karakteri sıfırladık.
    }
    return matris[ilkKelimeLength][ikinciKelimeLength]; // Matrisin en sağ
ve en altta olan indisi bize iki kelimenin birbirine Distance'ını verecek.
}


int main(){
    char word[50];
    baslangicKeyDegeri();   // Hash tablosunda başlangıçta tüm KEY
değerlerini -1 olarak işaretliyoruz.
    dokumanOku();  // smallDictionary sözlüğündeki kelimeler hash tablosuna
çekilir.
    while(1){
        char cumle[200];
        printf("\nCumle giriniz : ");
        fflush(stdin); // bilgisayarımda gets düzgün çalışmıyordu
internetten arattığımda bunun eklenmesi ile sorun çözülüyor yazıyordu ve
ekledim çalıştı.
        gets(cumle); // Girilecek cümle
        char temp[80]; // Cümleleri kelimelere ayırıp burada tutup bu dizi
üzerinden işlem yapacağız.

        int len = strlen(cumle);
        int i = 0;
        while(i < len){  // Cümle karakter sayısı kadar döngü kurulacak.
            int j = 0;
            while(i < len && cumle[i] != ' '){ // Her kelime arasında
boşluk var bu yüzden boşluk kontrolü yaptık.
```

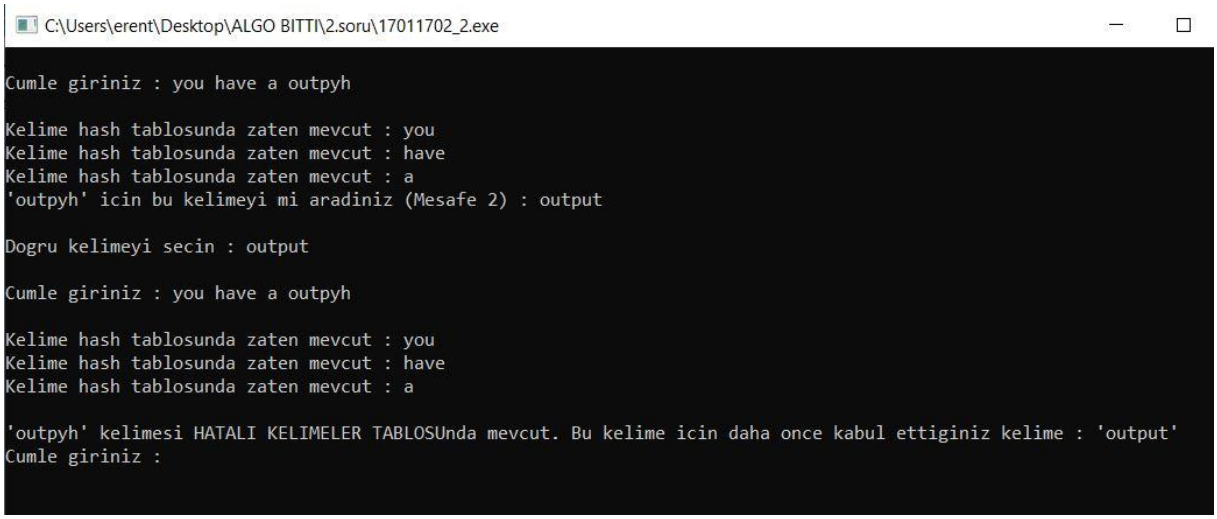
```
                temp[j] = cumle[i];  // Cümledeki her karakteri temp'e
atıyoruz.
                j++;
                i++;
            }
            i++;
            int key; // hatali kelimenin key değeri
            key = hornerMetodu(temp); // kelime icin key değeri üretiyoruz
horner fonksiyonu ile
            if(hataliTablodaArama(key,temp) == 1){ // Daha önceden hatalı
tabloda varmı ?
                // print islemini   hataliTablodaArama() icinde yaptıgım
icin buradan işlem yapmama gerek kalmadı.
            }
            else{
                char dogruKelime[50]; // onerilen kelimeler için
gireceğimiz dogru kelime
                switch(kelimeAra(temp)){  // kelime arama fonksiyonuyla
gerçekleşecek olan işlem üzerinden dönen return değerine göre switch
kontrolü yapıyoruz.
                    case 1: break;  // Kelime hash tablosunda mevcut
çıktısı verir.
                    case 2: printf("\n'%s' kelimesi HASH TABLOSUnda yok
ayrica onerilen kelimede bulunamadi !",temp); break;
                    default :
                            printf("\n\nDogru kelimeyi secin : ");
                            scanf("%s",dogruKelime);
                            if(dogruKelimeyiAra(dogruKelime) == 1){  //
Onerilenlere göre kullanıcının girdiği dogru kelimeyi yeniden hash
tablosunda arıyoruz.
                                ekleHataliTablo(key,temp,dogruKelime); //
varsa onerilenlerden doğru kelime girmiştir ve hatali tabloya ekleme
yapıyoruz.
                            }
                            else{
                                printf("\nOnerilen kelimelerin disinda bir
secim yaptiniz !(Cunku hash tablosunda girdiginiz kelime yok !)\n");
                            }
                            break;
                }
            }
            for(j=0;j<80;j++){  // kelimeyi tuttugum temp'i bir sonraki
kelimeyi alacağımdan dolayı temizliyorum.
                temp[j] = '\0';
            }
        }
    }

    return 0;
}
```

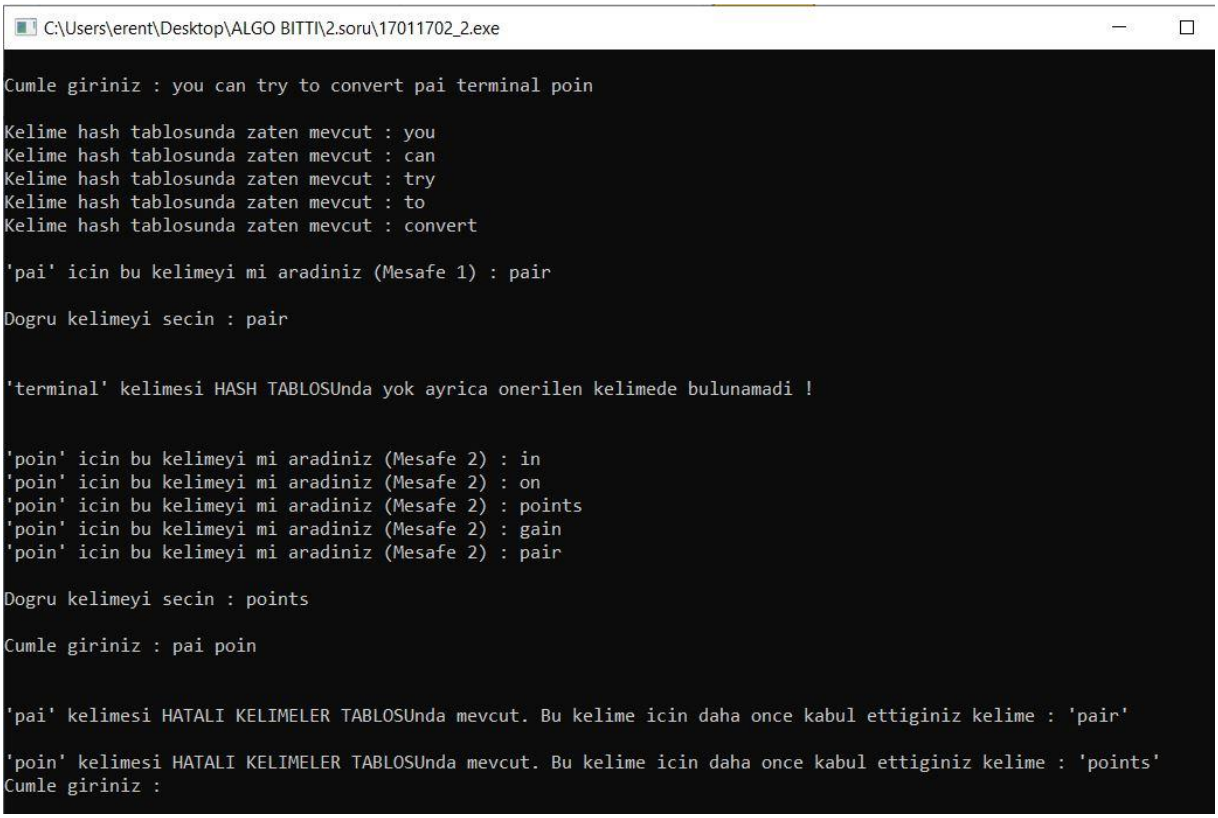
EKRAN ÇIKTILARI


C:\Users\erent\Desktop\ALGO BITTI\2.soru\17011702_2.exe
Cumle giriniz : It is the samd
Kelime hash tablosunda zaten mevcut : it
Kelime hash tablosunda zaten mevcut : is
Kelime hash tablosunda zaten mevcut : the
'samd' icin bu kelimeyi mi aradiniz (Mesafe 1) : same
Dogru kelimeyi secin : same
Cumle giriniz : samd
'samd' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu kelime icin daha once kabul ettiginiz kelime : 'same'
Cumle giriniz :


C:\Users\erent\Desktop\ALGO BITTI\2.soru\17011702_2.exe
Cumle giriniz : you are goo
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : are
'goo' icin bu kelimeyi mi aradiniz (Mesafe 1) : good
Dogru kelimeyi secin : good
Cumle giriniz : you are goo
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : are
'goo' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu kelime icin daha once kabul ettiginiz kelime : 'good'
Cumle giriniz :


C:\Users\erent\Desktop\ALGO BITTI\2.soru\17011702_2.exe
Cumle giriniz : you have homewofl
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : have
'homewofl' icin bu kelimeyi mi aradiniz (Mesafe 2) : homework
Dogru kelimeyi secin : homework
Cumle giriniz : you have homewofl
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : have
'homewofl' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu kelime icin daha once kabul ettiginiz kelime : 'homework'
Cumle giriniz :


C:\Users\erent\Desktop\ALGO BITTI\2.soru\17011702_2.exe
Cumle giriniz : you have a outpyh
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : have
Kelime hash tablosunda zaten mevcut : a
'outpyh' icin bu kelimeyi mi aradiniz (Mesafe 2) : output
Dogru kelimeyi secin : output
Cumle giriniz : you have a outpyh
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : have
Kelime hash tablosunda zaten mevcut : a
'outpyh' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu kelime icin daha once kabul ettiginiz kelime : 'output'
Cumle giriniz :


C:\Users\erent\Desktop\ALGO BITTI\2.soru\17011702_2.exe
Cumle giriniz : you can try to convert pai terminal poin
Kelime hash tablosunda zaten mevcut : you
Kelime hash tablosunda zaten mevcut : can
Kelime hash tablosunda zaten mevcut : try
Kelime hash tablosunda zaten mevcut : to
Kelime hash tablosunda zaten mevcut : convert
'pai' icin bu kelimeyi mi aradiniz (Mesafe 1) : pair
Dogru kelimeyi secin : pair
'terminal' kelimesi HASH TABLOSUnda yok ayrica onerilen kelimede bulunamadi !
'poin' icin bu kelimeyi mi aradiniz (Mesafe 2) : in
'poin' icin bu kelimeyi mi aradiniz (Mesafe 2) : on
'poin' icin bu kelimeyi mi aradiniz (Mesafe 2) : points
'poin' icin bu kelimeyi mi aradiniz (Mesafe 2) : gain
'poin' icin bu kelimeyi mi aradiniz (Mesafe 2) : pair
Dogru kelimeyi secin : points
Cumle giriniz : pai poin
'pai' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu kelime icin daha once kabul ettiginiz kelime : 'pair'
'poin' kelimesi HATALI KELIMELER TABLOSUnda mevcut. Bu kelime icin daha once kabul ettiginiz kelime : 'points'
Cumle giriniz :