Ders Adı: Algoritma Analizi
Öğrenci Bilgileri: Ömer Buğrahan Çalışkan – 17011076
Ödev İçeriği: Sorgulanan bir cümlede yanlış yazılmış kelimeler varsa bu kelimelerin yerine doğru kelimeler öneren bir sistem tasarımı.

Kod Açıklaması:

Sozluk Structure: Kelime bilgileri structure yapısında tutuluyor. Key horner metodu sonucu oluşan değeri tutuyor, value kelimenin kendisini, oneri ise hatalı bir kelime girildiyse bu kelimenin yerine önerilecek kelimeyi tutuyor.

FindKey Fonksiyonu: Fonksiyon, okunan kelimeyi ve hash tablosunu alıyor. Alınan kelimenin key karşılığı horner metodu ile hesaplanır ve return ile döndürülür.

DoubleHash Fonksiyonu: Key değeri hesaplanacak structure nesnesi ve hashing formülü için gereken i sayısı alınır. Double Hashing formülü sonucu oluşan adres değeri hesaplanır.

InserToHash Fonksiyonu: Structure nesnesi ve hash tablosu alınır. İ değeri 0'dan başlanarak doublehashing yöntemiyle adres aranır eğer adres boş ise adresin içi doldurulur. Boşta hiçbir yer yok ise Kelime Eklenemedi bilgisi verilir.

Readfile Fonksiyonu: Hash Tablosu alınır. Oluşturulacak hash dosyasının kelimeleri dosyadan okunur, fscanf fonksiyonu ile her bir kelime önce structure nesnesi olarak oluşturulur ardından gerekli bilgileriye hash tablosuna eklenir ve dosya kapatılır.

SearchToHash Fonksiyonu: Structure nesnesi ve hash tablosu alınır. Aranan kelime double hashing yöntemiyle ile aranır. Eğer kelime tabloda var ise adresi döndürülür yok ise 0 değeri döndürülür.

Distance Fonksiyonu: 2 adet kelime ve bu kelimelerin uzunlukları alınır. Dinamik programlama kullanılarak 2 kelimenin birbirine olan uzaklığı hesaplanır ve bu değer matrisin son elemanı olarak döndürülür.

Offerword Fonksiyonu: Aranan kelime, hash tablosu, error tablosu, structure nesnesi, istenilen distance değeri ve ekrana en son basılacak doğru cümle dizisi alınır. Kelimenin sözlükte olmadığı belirtilir ve hash tablosunda içi dolu olan bütün adreslerdeki kelimelerin, aranan kelimeyle olan uzaklığı hesaplanır ve istenilen distance değeri ile karşılaştırılır, eğer eşit ise bu kelime ekrana bastırılır ve index dizisine adres değeri atılır. Bu döngü bittikten sonra kullanıcıya hangi kelimeyi seçmek istediği sorulur, kullanıcı önerilen kelimelerden birini girmez ise cevabı hatalı kabul edilir ve tekrar seçilmesi istenir. Seçiminin yapılmasının

ardından önerilen kelime hash tablosuna eklenir ve en son yazılacak doğru cümle stringine atılır.

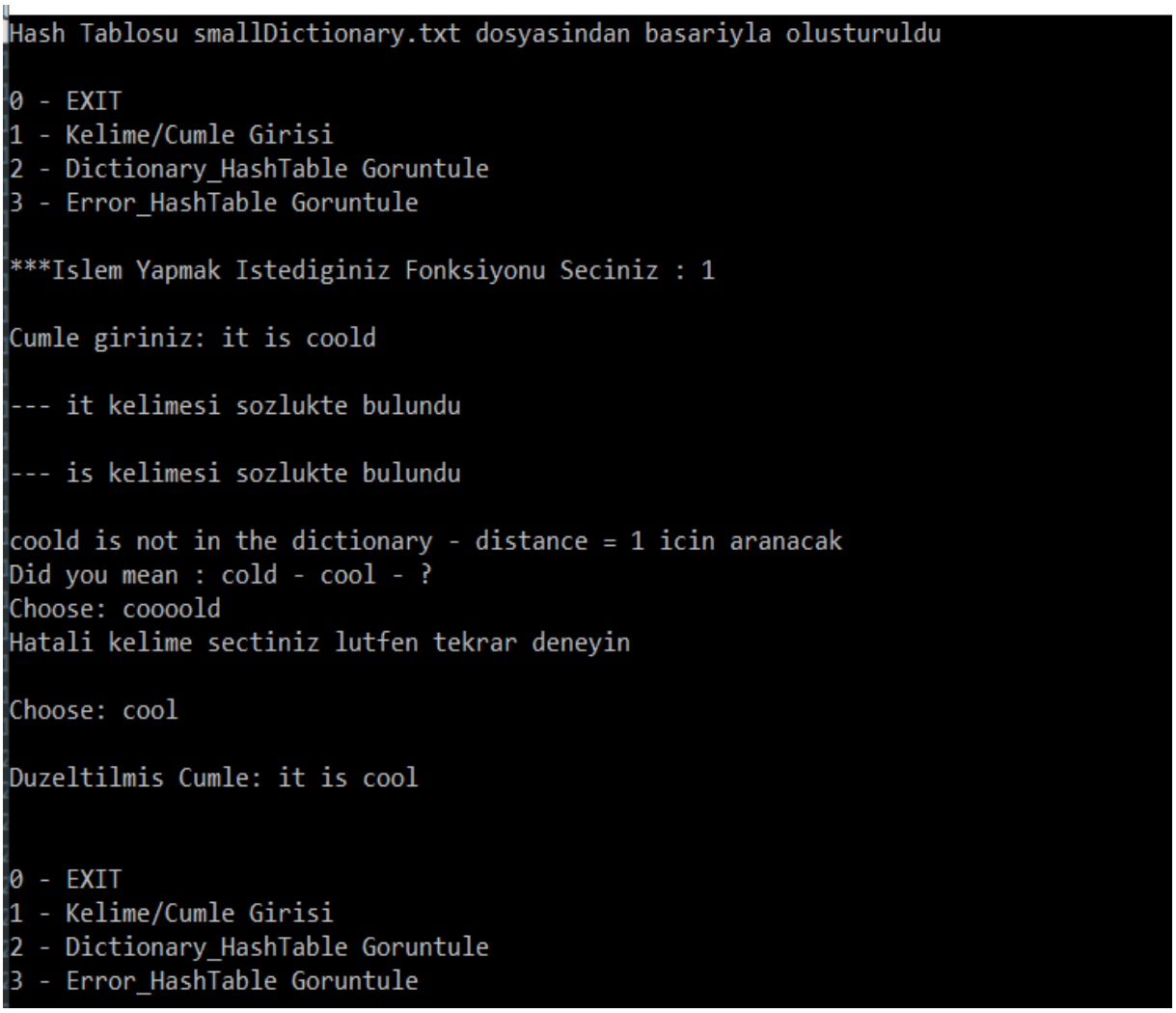
MenuYazdır: Menu Fonksiyonu.

Main Fonksiyonu: Menu gösterilir ve yapılmak istenen fonksiyon sorulur.

1) Kelime/Cümle Girisi: Girilen cümle kelime kelime bölünerek (strtok) hash tablosunda aranır, eğer yok ise error table da aranır orda da yok ise distance=1 için offerword fonksiyonu çağırılarak yeni kelimeler önerilir var ise seçenekler kullanıcıya sorulur ve doğru cümlenin devamına eklenir, yok ise distance=2 için denenir aynı şekilde kullnıcıya sorulur yok ise daha büyük distance değerlerine bakılmayacağı için kelime atlanır ve bu bilgi kullanıcıya söylenir. Kelime error table da bulunur ise yanlış yazılmış hali ve önceden belirlenmiş önerilen hali kullanıcıya söylenir ve bu hali doğru cümleye eklenir. Kelime sözlükte bulunur ise sözlükte bulunduğu bilgisi kullanıcıya söylenir ve doğru cümleye ekleme yapılır. Bu şekilde girilen cümledeki kelimeler bitene kadar döngü devam eder cümle tamamen bittiğinde yazılması gereken doğru cümle ekrana bastırılır ve tekrar menü gösterilerek yapılmak istenen fonksiyon sorulur.
2) Dictionary_Hashtable Goruntule: smallDictionary.txt dosyasından okunan kelimelerin yerleştirildiği hash tablosu görüntülenir.
3) Error_Hashtable Goruntule: Dictionary Hash Tablosunda olmayan kelimelere karşılık kullanıcı tarafından girilmiş önerilerin ve kelimelerin kendisinin tutulduğu hash tablosu görüntülenir.
4) EXIT: 0 girildiginde veya 1,2,3 dısında bir değer girildiğinde program sonlanır.

Ekran Görüntüleri:

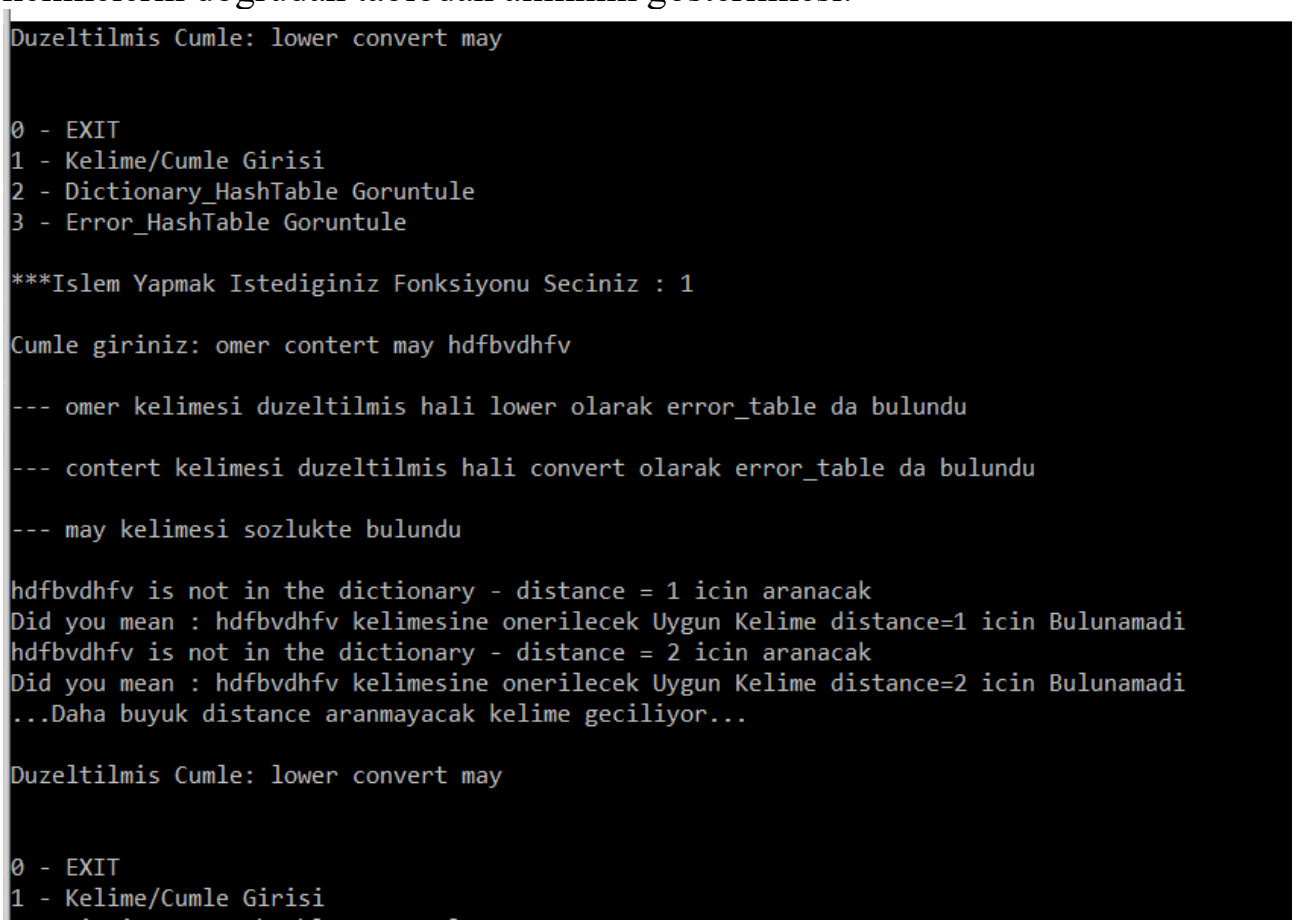
1) Örnek olarak verilen cümlenin denenmesi ve önerilen kelimelerin dışında bir kelime girildiğinde kabul etmemesinin gösterilmesi.



2) Sozlukte bulunan kelime, distance=1 icin önerilecek kelimesi olan kelime, distance=1 icin önerilecek kelimesi olmayan fakat distance=2 icin önerilecek kelimesi olan kelime ve hem distance=1 hem distance=2 icin önerilecek kelimesi olmayan cümlenin gösterilmesi.

```
0 - EXIT
1 - Kelime/Cumle Girisi
2 - Dictionary_HashTable Goruntule
3 - Error_HashTable Goruntule

***Islem Yapmak Istediginiz Fonksiyonu Seciniz : 1

Cumle giriniz: omer contert may hdfbvdhfv

omer is not in the dictionary - distance = 1 icin aranacak
Did you mean : omer kelimesine onerilecek Uygun Kelime distance=1 icin Bulunamadi
omer is not in the dictionary - distance = 2 icin aranacak
Did you mean : one - other - or - lower - ?
Choose: lower

contert is not in the dictionary - distance = 1 icin aranacak
Did you mean : convert - ?
Choose: convert

--- may kelimesi sozlukte bulundu

hdfbvdhfv is not in the dictionary - distance = 1 icin aranacak
Did you mean : hdfbvdhfv kelimesine onerilecek Uygun Kelime distance=1 icin Bulunamadi
hdfbvdhfv is not in the dictionary - distance = 2 icin aranacak
Did you mean : hdfbvdhfv kelimesine onerilecek Uygun Kelime distance=2 icin Bulunamadi
...Daha buyuk distance aranmayacak kelime geciliyor...

Duzeltilmis Cumle: lower convert may
```

3) Örnek 2’de girilen cümlenin aynısının girilmesiyle bu sefer önerilecek kelimelerin doğrudan tablodan alımının gösterilmesi.



4) Dictionary Table’ın gösterilmesi.

```
0 - EXIT
1 - Kelime/Cumle Girisi
2 - Dictionary_HashTable Goruntule
3 - Error_HashTable Goruntule

***Islem Yapmak Istediginiz Fonksiyonu Seciniz : 2
0 - prefer
1 - understand
4 - tooth
7 - dynamically
8 - assume
9 - below
10 - traditionally
12 - low
13 - options
14 - load
15 - compiled
16 - any
17 - forth
18 - you
19 - size
20 - how
22 - need
23 - possible
24 - spell
25 - input
28 - simple
29 - cannot
31 - functions
```

5) Örnek 2’de yapılan işlemler sonucu oluşan Error Table’ın ekrana yazdırılmasının gösterilmesi.

```
0 - EXIT
1 - Kelime/Cumle Girisi
2 - Dictionary_HashTable Goruntule
3 - Error_HashTable Goruntule

***Islem Yapmak Istediginiz Fonksiyonu Seciniz : 3
1 - contert - convert
425 - omer - lower
```

C Kodu:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define m 500

typedef struct sozluk{ // İcinde kelimenin degeri, keyi ve onerilen kelimeyi tutan structure
   int key;
   char value[50];
   char oneri[50];
}ht;

int FindKey(char buffer[50], ht table[m]){ // Horner metodu kullanılarak kelime degerinin key
hesaplanması
   int word_length = strlen(buffer);   // kelimenin uzunluğu
   int key=0,i=0;

      for(i=0;i<word_length;i++)                // Horner Metodu ile key hesaplanması
         key += buffer[i]*pow(3,i);
   return key;
}

int doubleHash(ht data,int i){// Double hashing yöntemiyle adres hesaplanması
         int h1 = data.key % m;
   int h2 = 1 + (data.key%(m-1));
   int a = (h1 + i*h2) % m;
   return a;
}

int InsertToHash(ht data,ht table[m]){          // Double hashing kullanılarak key değerinin hash
tablosuna yerlestirilmesi
   int i=0;
   int adr = doubleHash(data,i);

   while((table[adr].key!=-1) && (i < m)){  // Tabloda boşluk bulunana kadar yeni adres hesaplanır
                  i++;
      adr = doubleHash(data,i);
   }
         if(table[adr].key == -1){        // Adresteki yer boş ise yerleştirilme işlemi, eğer error table ise
onerilen kelime de yerleştirilir.
                  table[adr].key=data.key;
         strcpy(table[adr].value,data.value);
         strcpy(table[adr].oneri,data.oneri);
         return adr;
                  }
                  printf("Kelime Eklenemedi: %s\n",data.value);
                  return -1;
```

```
}

void readfile(ht table[m]){
          int i, adr;
          FILE* sozluk;
   sozluk = fopen("smallDictionary.txt","r");           // Okunacak dosya
      for(i=0;i<m;i++){
          table[i].key=-1;          // Hash Tablosunun tamamının keylerine -1 atanır.
          }
      if (sozluk == NULL){
          printf("Dosya acilamadi.");
             }
                   else{
          printf("Hash Tablosu smallDictionary.txt dosyasindan basariyla olusturuldu\n");
          char buffer[50];
          while (fscanf(sozluk, "%s", buffer) > 0){     // hash dosyasının kelime kelime okunması -
smallDictionary.txt

             ht data;                          // Kelimenin structure a keyi ile beraber eklenmesi
             data.key=FindKey(buffer,table);
             strcpy(data.value,buffer);
             adr=InsertToHash(data,table);
                }
             }
   fclose(sozluk);
}

int SearchToHash(ht data,ht table[m]){
   int i=0;       // Aranan kelimenin double hashing algoritması ile bulunması
   int adr = doubleHash(data,i);

          while((i < m)){                    // Eğer tabloda bulunursa adres dondurulur yoksa 0 dondurulur
                   if(strcmp(table[adr].value,data.value)==0){
      return adr;
          }
                   i++;
      adr = doubleHash(data,i);
   }
      return 0;
}

int distance (char * word1, int len1, char * word2, int len2){          // İki kelimenin uzaklıgının
karsılastırılması
   int matrix[len1 + 1][len2 + 1];
   int i;
   for (i = 0; i <= len1; i++) {// İlk satır ve ilk sütun doldurulur
      matrix[i][0] = i;
   }
   for (i = 0; i <= len2; i++) {
      matrix[0][i] = i;
```

```
    }
    for (i = 1; i <= len1; i++) {// Kelimelerin matriste karşılık gelen harflerine göre minimum uzaklık
hesplanır
        int j;
        char c1;

        c1 = word1[i-1];
        for (j = 1; j <= len2; j++) {
            char c2;

            c2 = word2[j-1];
            if (c1 == c2) {
                matrix[i][j] = matrix[i-1][j-1];       // Copy ise sol üstten cost olmadan tasınır.
            }
            else {
                int deleted;
                int insert;
                int substitute;
                int minimum;

                deleted = matrix[i-1][j] + 1;                   // Delete ise soldan sağa 1 arttırılarak tasınır
                insert = matrix[i][j-1] + 1;             // İnsert ise yukarıdan aşağıya 1 arttırılarak tasınır
                substitute = matrix[i-1][j-1] + 1;  // Change ise sol üstten 1 arttırılarak tasınır.
                minimum = deleted;                       // En kucuk olan alınacagı icin minimum degerine delete
atılır ve diger degerlerle karsılastırılır
                if (insert < minimum) {
                    minimum = insert;
                }
                if (substitute < minimum) {
                    minimum = substitute;
                }
                matrix[i][j] = minimum;    // En son bulunan minimum degeri tabloya yazılır
            }
        }
    }
    return matrix[len1][len2];  // Bulunan distance degeri son hücrede olusur ve bu deger dondurulur.
}

int offerword(char aranan[50], ht table[m], ht errtable[m], ht data, int distsize, char sentence[200]){
          int i,count=0,dist;
          int index[50];

          printf("\n%s is not in the dictionary - distance = %d icin aranacak \nDid you mean :
",aranan,distsize); // Sozluk Tablosunda bulunamayan deger aranacak distance degeri ile birlikte
kullanıcıya söylenir
        for(i=0;i<m;i++){
                            if(table[i].key != -1){
                                    dist=distance(aranan,strlen(aranan),table[i].value,strlen(table[i].value));
// Sozluk tablosunda bulunan her kelimenin aranan deger ile uzaklıgı bulunur
                                            if(dist == distsize){
```

```
                                                printf("%s - ",table[i].value); // Eger bulunan uzaklık
bizim aradıgımız 1 veya 2 degerine esit ise kullanıcıya onerilir
                                                index[count++] = i;    // index dizisine bu adresin degeri
yazılır
                                                }
                                        }
                                }
                if(count==0)
                        return 0;       // Eger aradıgımız uzaklıkta bir kelime yok ise count
artmayacaktır ve bu durumda 0 degeri dondurulecektir

                printf("?");

                int flag=0;
                char temp[50];

                while(flag==0){
                        printf("\nChoose: ");  // Kullanıcıdan oneri olarak girebilecegi kelimelerden
birini secmesi beklenir
                        scanf("%s",temp);               // Kullanıcının yazdıgı oneri

                for(i=0;i<count;i++){ // Kullanıcının yazdıgı oneri, distance tarafından bulunan
onerilenlerle karsılastırılır eger esit ise dongu sonlandırılır fakat hatalı bir oneride bulunur ise
tekrardan secmesi istenir
                        if( strcmp(table[index[i]].value,temp)==0 ){
                                flag=1;
                                break;
                        }
                }
                if(flag==0)
                        printf("Hatali kelime sectiniz lutfen tekrar deneyin\n");
                }

                strcpy(data.oneri,temp);        // Dogru oneri secilir ise bu oneri error table a structure
nesnesinin onerisi olarak eklenir
                InsertToHash(data,errtable);
                strcat(sentence,data.oneri);    // En son gosterilecek dogru cumleye bu onerilen kelime
eklenir
                strcat(sentence," ");

                return 1;
}

void Menu_Yazdir(){                     // Menu Yazdıran fonksiyon
        printf("\n0 - EXIT\n");
        printf("1 - Kelime/Cumle Girisi\n");
        printf("2 - Dictionary_HashTable Goruntule\n");
        printf("3 - Error_HashTable Goruntule\n");
}
```

```
int main(){
	int i;				// Dongu degiskeni
	ht table[m];		// Sozluk Hash Table
	ht errtable[m];		// Error Hash Table
	char satir[200];	// Butun satırı okuyacak char dizisi
	char *aranan;		// Aracak olan kelime
	int errorflag;		// Kelimenin sozlukte olup olmadıgını tutacak olan degisken
	int distselect;		// Offerword fonksiyonundan 0 degeri donerse yani istenilen distance
icin kelime onerilemezse bu durumu tutacak olan degisken
	int secim;			// Menu secimi icin kullanılacak degisken
	char sentence[200];	// En son yazdırılacak dogru cumleyi tutan char dizisi

	for(i=0;i<m;i++){
	errtable[i].key=-1;	// Olusturulan Error Hash Table'ın butun key degerlerine -1 verilir
	}

	readfile(table);	// dictionary file okunur ve tabloya eklenir

	Menu_Yazdir();
	printf("\n***Islem Yapmak Istediginiz Fonksiyonu Seciniz : ");
	scanf("%d", &secim);

	do{
		switch(secim){
			case 1:
				sentence[0]='\0';	// Her yeni cumle girisinde dogru cumlenin
sıfırlanması gerekli
				printf("\nCumle giriniz: ");
				fflush(stdin);	// Buffer da kalan degerlerin temizlenmesi ve gets
fonksiyonunun duzgun calısması icin gereken fonksiyon
				gets(satir);
				aranan = strtok(satir," ");	// Butun satır okunur ve strtok fonksiyonu
ile bosluklarla bölünür
			int dist;

				while(aranan != NULL){
				ht data;
			data.key = FindKey(aranan,table);
			strcpy(data.value,aranan);
			errorflag = SearchToHash(data,table); // Aranacak olan kelimenin
SearchToHash fonksiyonundan dondurdugu deger errorflag degiskenine atanır
			if(errorflag == 0){	// 0 ise sozluk tablosunda yoktur
				if(SearchToHash(data,errtable) == 0){	// Bu sefer errortable
da aranır burada da yok ise
					distselect =
offerword(aranan,table,errtable,data,1,sentence);	// distance=1 icin kelimeler onerilir
					if(distselect == 0){	// Onerilecek kelime
yok ise bu sefer distance=2 icin aranır
						printf("%s kelimesine onerilecek
Uygun Kelime distance=1 icin Bulunamadi",aranan);
```

```
                                                    distselect =
offerword(aranan,table,errtable,data,2,sentence);
                                                        if(distselect == 0){     //
Eger distance=2 icin de kelime onerilemediyse daha fazla distance icin aranmaz, kelime gecilerek
kullanıcıya bilgi verilir
                                                            printf("%s
kelimesine onerilecek Uygun Kelime distance=2 icin Bulunamadi\n",aranan);
                                                            printf("...Daha
buyuk distance aranmayacak kelime geciliyor...\n");
                                                        }
                                                    }
                                                }
                                                else{    // Error Table da aranan kelime onceden
onerilip bulunduysa kullanıcıya gosterilir, adresi hesaplanarak dogru cumleye onerilen kelime eklenir
                                                    int adrs = SearchToHash(data,errtable);
                                                    printf("\n--- %s kelimesi duzeltilmis hali
%s olarak error_table da bulundu\n",aranan,errtable[adrs].oneri);
                                                    strcat(sentence,errtable[adrs].oneri);
                                                    strcat(sentence," ");
                                                }
                                            }
                                            else{    // Kelime sozlukta var ise kullanıcıya bilgi verilir ve
dogru cumleye eklenir
                                                printf("\n--- %s kelimesi sozlukte bulundu\n",aranan);
                                                strcat(sentence,aranan);
                                                strcat(sentence," ");
                                            }
                                        aranan = strtok(NULL," "); // Bosluga kadar yeni kelime alınır.
                                            }
                printf("\nDuzeltilmis Cumle: %s\n\n",sentence); // Yapılan islemler sonucu olusan
dogru cümle ekrana yazdırılır

                Menu_Yazdir();
                printf("\n***Islem Yapmak Istediginiz Fonksiyonu Seciniz : ");
                scanf("%d", &secim); // Kullanıcıya yapmak istedigi yeni islem sorulur.
        break;
                case 2:
                    for(i=0;i<m;i++){      // Dictionary Tablosunun dolu olan degerleri gosterilir
                        if(table[i].key!=-1)
                            printf("%d - %s\n",i,table[i].value);
                        }
                    Menu_Yazdir();
                    printf("\n***Islem Yapmak Istediginiz Fonksiyonu Seciniz : ");
                    scanf("%d", &secim); // Kullanıcıya yapmak istedigi yeni islem sorulur.
        break;
                case 3:
                    for(i=0;i<m;i++){      // Error Tablosunun dolu olan degerleri gosterilir
                        if(errtable[i].key!=-1)
                            printf("%d - %s - %s\n",i,errtable[i].value,errtable[i].oneri);
                        }
```

```
                        Menu_Yazdir();
                        printf("\n***Islem Yapmak Istediginiz Fonksiyonu Seciniz : ");
                        scanf("%d", &secim); // Kullanıcıya yapmak istedigi yeni islem sorulur.
                        break;
        }
}while(secim == 1 || secim == 2 || secim == 3);
        printf("Program Sonlandi...");
}
```