Course Name: VERİ YAPILARI VE ALGORİTMALAR
Course Group: Group 1
Instructor Name: Doç. Dr. M.Elif KARSLIGİL

ÖDEV 3

Student Id: 16011706
Student Name and Surname: Duygu Erduran

Konu: Find and Replace Uygulaması

Problem: Bu ödevde, Boyer-Moore Horspool algoritması kullanarak verilen bir kelimeyi bir metin içerisinde arayan ve bulduğu yerlerde gerekli değişikliği yapan bir algoritma tasarlamanız ve kodunu C dilinde yazmanız istenmektedir. Belirtilen uygulama bir editör'de kullanılan "Find and Replace" mantığı ile çalışacaktır. Geliştireceğiniz algoritma kullanıcıya "Case Sensitive" seçeneği sunmalıdır. Özellik seçilerek küçük büyük harf farkı aktif hale getirilebilmelidir. Bu özellik seçilmediyse algoritma küçük büyük harf ayrımı gözetmeksizin arama gerçekleştirmelidir.

Açıklama:

İstenilen girdiler yapılmaya çalışıldı. Lakin Hassiyet ayarı kullanıcıya sorulmadı. Program Hassasiyet açık konumda çalışamadığı için sadece Hassasiyet kapalı olarak işlevdedir.

C Kodu :

```
#include<stdio.h>
#include<stdlib.h>
#include <windows.h>
#include <locale.h>
#include<string.h>
#include<time.h> //clock fonksiyonunu kullanmamizi saglar
#define MAX 2048 // Metinin maksimum alabilecegi degeri tanimladik

void dosya_okuma(char metin[MAX]);// dosyayi okuma fonksiyonu
void shiftTable(char Bul[], int table[], int boyut); //  Boyer-Moore
Algoritmasi shift table olusturma fonksiyonu
void arama(char Bul[], char Metin[], int* index_dizisi) ; // arama
yapan fonksiyon
void replace_islemi(char* metin, char* bul, char* yeni, int* index);
// replace islemini yapan fonksiyon

void dosya_okuma(char metin[MAX]){
```

```
    char dosya_ismi[30];
    printf("'duygu.txt ' dosyasýný çalýþtýrnýz. ");
    scanf("%s",dosya_ismi);
    FILE *fp = fopen(dosya_ismi,"r"); //dosyayi read modunda acariz
    int i = 0;
    while(!feof(fp)){   // dosya sonuna kadar karakterler okunur.
        metin[i] = getc(fp);
        i++;
    }
    metin[i-1] = '\0';    //Dizinin son elemani string oldugu icin  \0
koyar.
    fclose(fp);
}
void shiftTable(char Bul[], int table[], int boyut)
{ // shif table olusturulmasi icin gerekli döngüler
    int i;
    for (i = 0; i < MAX; i++)
        table[i] = -1; //table  -1 ile doldurur
    for (i = 0; i < boyut; i++)
        table[(int) Bul[i]] = i;
}
void arama(char Bul[], char Metin[], int* index_dizisi)
{   int m = strlen(Bul);    // degiskenlerimizi tanimladik.
    int n = strlen(Metin); //Aramaya her defasinda 0'dan baslamak
yerine bir önceki bulunan indisin bir sonrasindan baslanmamiz lazimdi.
    int i = 0,k = 0, skip = 0,j,table[MAX];
    shiftTable(Bul, table, m); // shift yapan fonksiyonu cagirdik
    int uzunluk = n-m;
    while(i <= uzunluk){
        j = m-1;
         while ((j >= 0) && ( (Bul[j] == Metin[i+j]) || (abs(Bul[j] -
Metin[i+j]) == 32) ))
            {
                j--;
                //buyuk kucuk harf duyarli olmadigi icin ascii
tablosuna göre bu fark 32 dir.
            }
        if(j >= 0){

           if(Metin[i+j] < 92){
                skip = j - (table[Metin[i+j] + 32] >=
table[Metin[i+j]] ? table[Metin[i+j] + 32] : table[Metin[i+j]]);
                }
                else{
                     skip = j - (table[Metin[i+j] - 32] >=
```

```
table[Metin[i+j]] ? skip = table[Metin[i+j] - 32] :
table[Metin[i+j]]);
                }
            if(skip <= 0){
                skip = 1;
            }
        }
        else{ //eslesme bulunursa gerceklesir
            index_dizisi[k] =  i; // index degerlerini tutan dizi
            k++;
            if((i+m) < n){
                skip = m - table[Metin[i+j]]; //skip hesaplanir
                }
                else{
                      skip = 1;
                }
        }
        i += skip;
    }
    index_dizisi[k] =  -1;// dizinin  nerde bittigini anlamamizi
saglar
}
void replace_islemi(char* metin, char* bul, char* yeni, int* index)
{
    int yeni_boyut = strlen(yeni);
    int m = strlen(bul), n = strlen(metin),i,j,k,fark;

    if(yeni_boyut < m){  // yeni substring eskisinden daha küçük
oldugunda farki hesaplar.
        fark = m - yeni_boyut;
        for(i = 0; index[i] != -1; i++){
            for(j = index[i]+yeni_boyut - i*fark; j < n; j++){
                metin[j] = metin[j+fark];
            }
        }  // güncellik için metni kaydirir
        for(i = 0; index[i] != -1; i++){  // yeni substringi
yerlestirir.
            for(j = index[i] - i*fark, k = 0; k < yeni_boyut; j++,
k++){
                metin[j] = yeni[k];
            }
        }
    }
    else{ // eger yeni substring eskisinden büyükse
        fark = yeni_boyut - m;
        for(i = 0; index[i] != -1; i++){
```

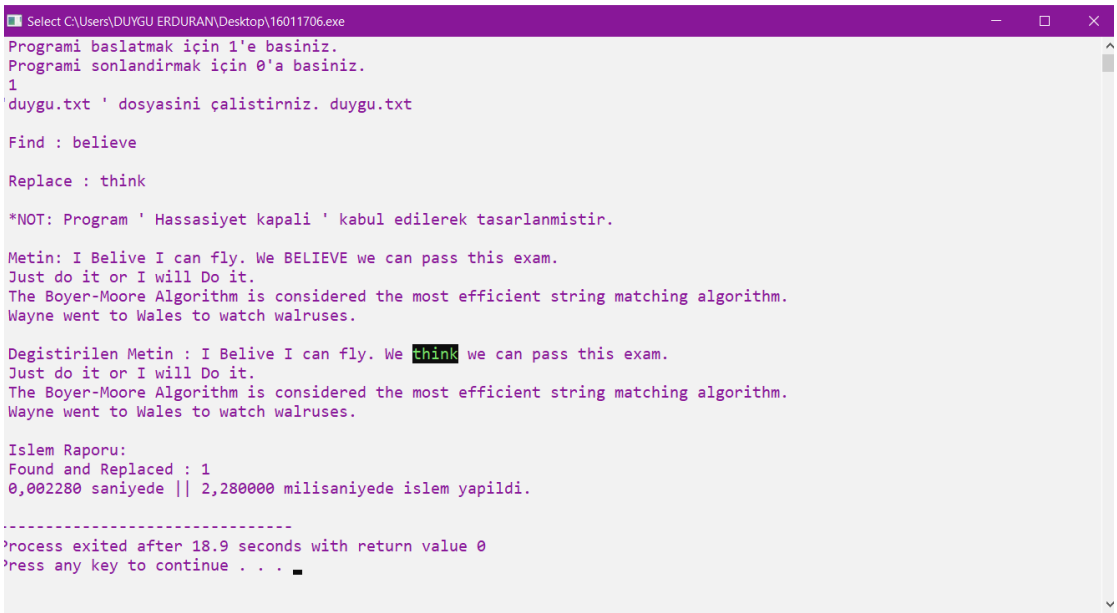
```
            for(j = n+fark; j >= index[i] + m; j--){
               metin[j] = metin[j-fark];
            }
            n = strlen(metin);
        }  // metin kaydirilir
        for(i = 0; index[i] != -1; i++){
            for(j = index[i] + i*fark, k = 0; k <yeni_boyut; j++,
k++){
                metin[j] = yeni[k];
            }
        }
        // yeni substring yerlestirilir.
    }
}
int main(){
    system(" color F5 "); // TEMA RENK AYARI
     setlocale(LC_ALL, "Turkish");  //TURKCELESTIRME ayari
    int i=0,tus;
    int index_dizisi[MAX];// substring indexlerini tutar
     char bul[MAX]; // aradigimiz substringi tutmak icin
    char metin[MAX]; //metin dosyasindakileri tutar
    char replace[MAX]; //yeni substring icin

    printf(" \n |Veri Yapýlarý Ödev-3 --> Find and Replace Uygulamasý
| \n\n");
    printf(" Programý baþlatmak için 1'e basýnýz.\n Programý
sonlandýrmak için 0'a basýnýz.\n ");
    scanf("%d",&tus);
    if(tus == 1)  // dosyayi okumaya baslatir
     dosya_okuma(metin);
     else if (tus == 0 ){ // programi sonlandirir.
     printf("Program sonlandý.");
     exit(0);
     }
    printf("\n Find : ");
    scanf("%s", bul);
    printf("\n Replace : ");
    gets(replace); // gets ile replace kelimelisini aliriz
    clock_t baslangic = clock(); // gecen süreyi hesaplamak icin islem
saatinin baslatiriz.
    fgets(replace,MAX,stdin); // fgets ile dosyadan raplace edilcek
kelimeyi aliriz
    strtok(replace,"\n"); // strtok fonksiyonu ile kelimeyi bulduktan
sonra devam ederken satir atlanmasi önlenir.bunun sebebi ise string
sonunda null olmasidir.
    printf(" \n *NOT: Program ' Hassasiyet kapalý ' kabul edilerek
```

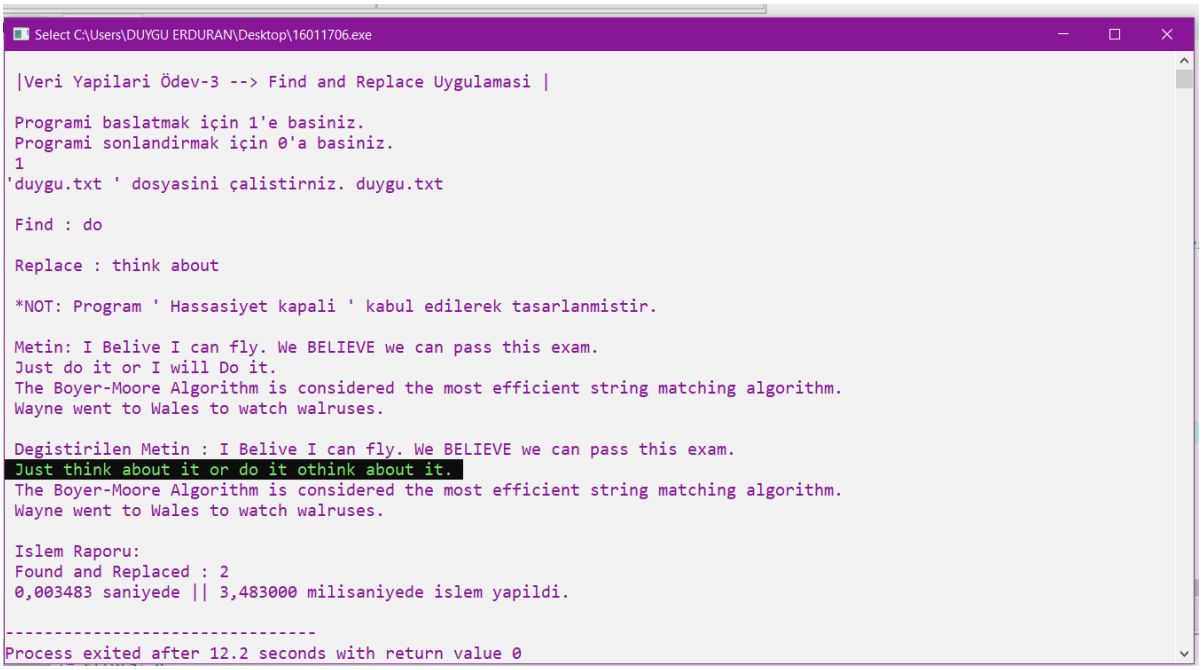
```
tasarlanmýþtýr.\n"); // bilgilendirme notu
    arama(bul,metin,index_dizisi); // arama fonksiyonu cagiriyoruz
    printf(" \n Metin: %s\n", metin); // degismeden önce metinimizi
yazdiriyoruzz.
      replace_islemi(metin,bul,replace,index_dizisi); // replace yapan
fonksiyonumuzu cagiriyoruz
    printf(" \n Degistirilen Metin : %s\n", metin); // degistirilen
metin yazdirilir
    while(index_dizisi[i] != -1){  // string bulunana kadar sayar.
      i++;
      }
      printf("\n Islem Raporu:\n Found and Replaced : %d\n",i);
//bulunan string yer degistirilmesinin sayisi yazdirilir.
    clock_t bitis= clock(); // islem saatini bitiririz.
    double mili_saniye= ((double)(bitis - baslangic) /
CLOCKS_PER_SEC); //milisaniyeyi hesaplar degiskene ataris
      printf(" %f saniyede || %f milisaniyede islem
yapýldý.\n",mili_saniye/1000.0, mili_saniye); //saniye ve milisaniyeyi
yazdiririz.
    return 0;
}
```

Analiz:

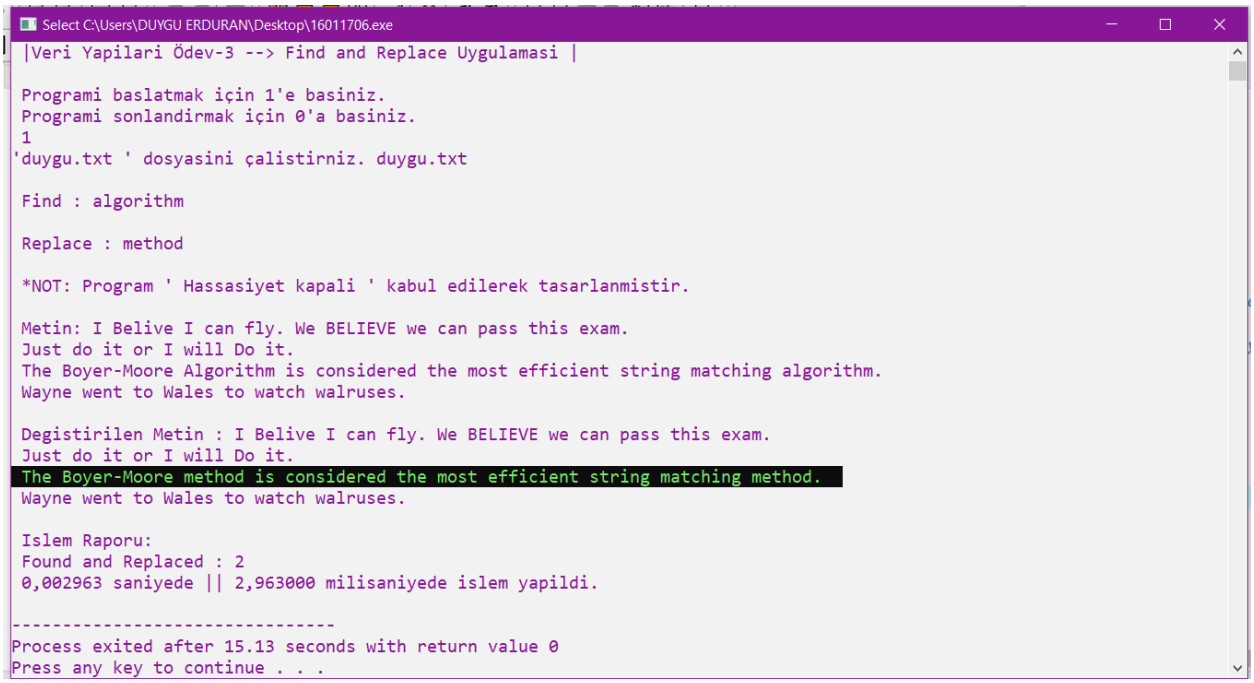
Örnek1:



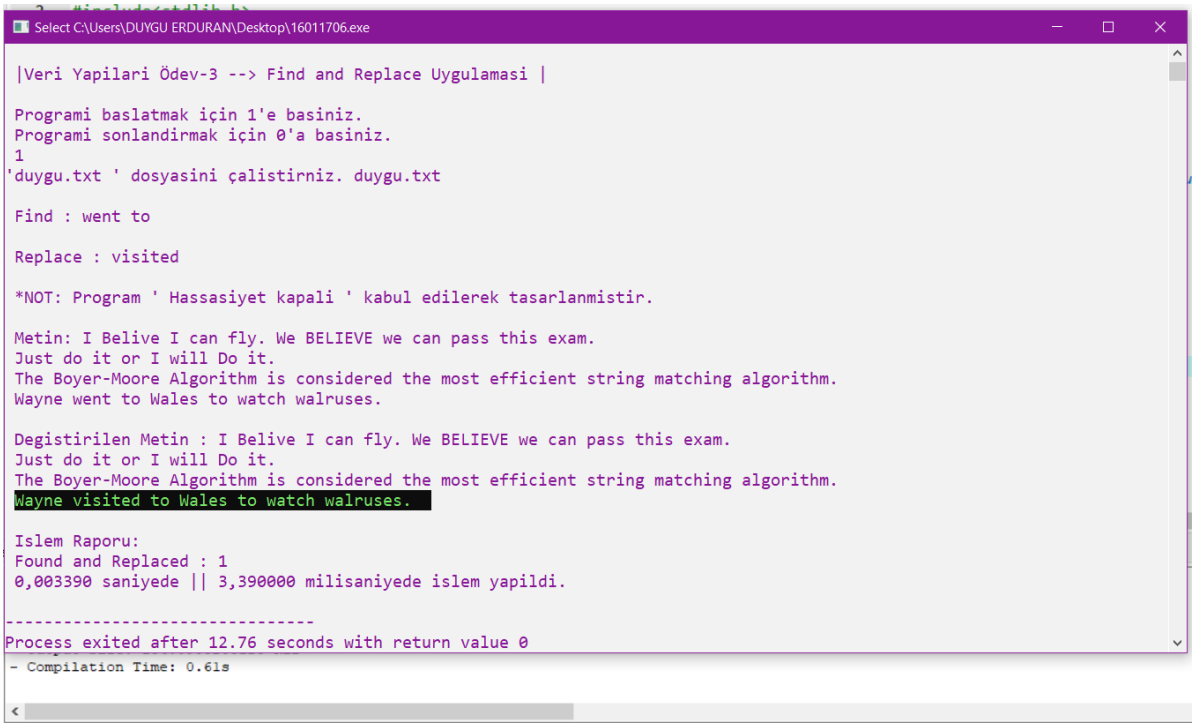
Örnek2:



Örnek3:



Örnek4:



Örnek5: