YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BLM3021 – ALGORİTMA ANALİZİ

DÖNEM SONU PROJE RAPORU

Konu : Kitap Öneri Sistemi

Ad Soyad : Eren TUTUŞ

Okul No : 17011702

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

Programın Çalışma Süreci

1-) bookRecommendation *createNode ( ) :

Her kullanıcı adını, o kullanıcıların hangi kitapları okuyup okumadığını ve hangi kitap için kaç puan verdiklerini tuttuğum struct yapısı için her gelen yeni kullanıcıya ait bilgiler için bu fonksiyonda bellekte yer açıyoruz.

2-) calculatedSimilarities *createNodeTwo ( ) :

Similarity değerlerinin tutulduğu struct yapısına gelecek olan her bilgi için bellekte yer açıyoruz.

3-) void enqueue (bookRecommendation *newInformation) :

Projede queue yapısı ile verileri struct yapısına aktardığım için dosyadan her çekilen satırı yani kullanıcı bilgilerini kuyruğa ekliyorum. Eğer kuyruğa ilk defa kullanıcı bilgileri eklenecekse front ve rear aynı düğümü işaret edecek. Bunun dışında her yeni gelen satır ile birlikte rear o satırı işaret edecek.

4-) void enqueue2 (calculatedSimilarities *newInformation) :

Yukarıdaki enqueue fonksiyonu ile aynı işlemi yapıyor fakat bunu 2.struct olan calculatedSimilarities yapısı için yapıyoruz. Similarity değerleri ile kullanıcı bilgilerini kuyruğa ekliyoruz.

5-) void readTheFile ( ) :

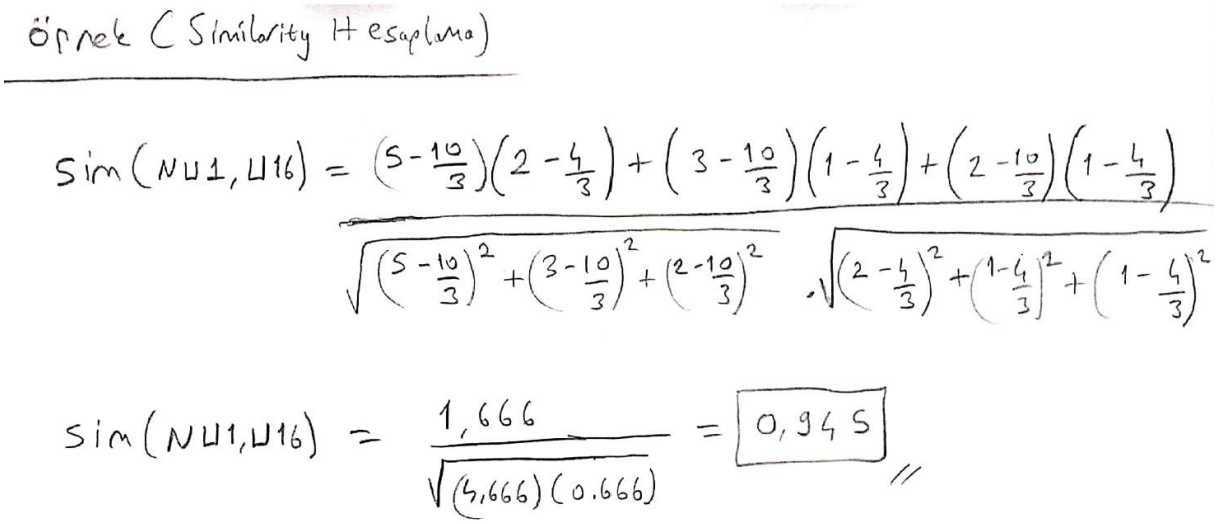
Bu fonksiyonda dosyadan verileri okuyoruz. Dosyadan verileri fgets ile satır satır okuyup o satırdan sütun bilgilerini ayrı ayrı alıyorum. İlk sütunda kullanıcı adı daha sonraki sütunlarda ise sırasıyla her kitap için verdikleri puanları alıyorum. Sütunlardaki her bilgi birbirinden noktalı virgül ile ayrıldığı için her puanı ve kullanıcı adını bu noktalı virgül karakteri üzerinden sorgu yaparak ayrıştırıyorum. Her satır bittiğinde ise bu verileri struct yapısına enqueue ediyorum.

6-) void calculateSimilarity (char userName [70]) :

Bir okuyucunun diğer okuyucuya olan benzerliğini (similarity) bu fonksiyonda hesaplıyoruz. Öncelikle parametre olarak gelen kullanıcı adını struct içinde var olup olmadığını doğrulamak

için döngü içinde koşul gerçekleştiriyoruz. Eğer var ise gerekli işlemlere başlıyoruz. Similarity değerlerini calculatedSimilarities struct yapısında tutacağımız için orada her kullanıcı için bellekte yer açıyoruz. Daha sonra U1…U20 kullanıcıları için parametre olarak gelen kullanıcı adı için similarity değerleri oluşturuyoruz. Bu değerleri oluştururken formülde ortak okudukları kitaplara verdikleri puan üzerinde değerlendirme yapıyoruz.

Örnek bir hesaplama :



7-) calculatedSimilarities *findKUsers (char userName [70]) :

Bir okuyucunun en benzer olduğu k kişinin belirlendiği fonksiyon. Parametre olarak gelen kullanıcı adının struct içinde var olup olmadığını doğrulamak için dönüyoruz. Similarityleri büyükten küçüğe doğru sıralayıp yerleştirdik. Yeni oluşan yapıyı döndürüyoruz.

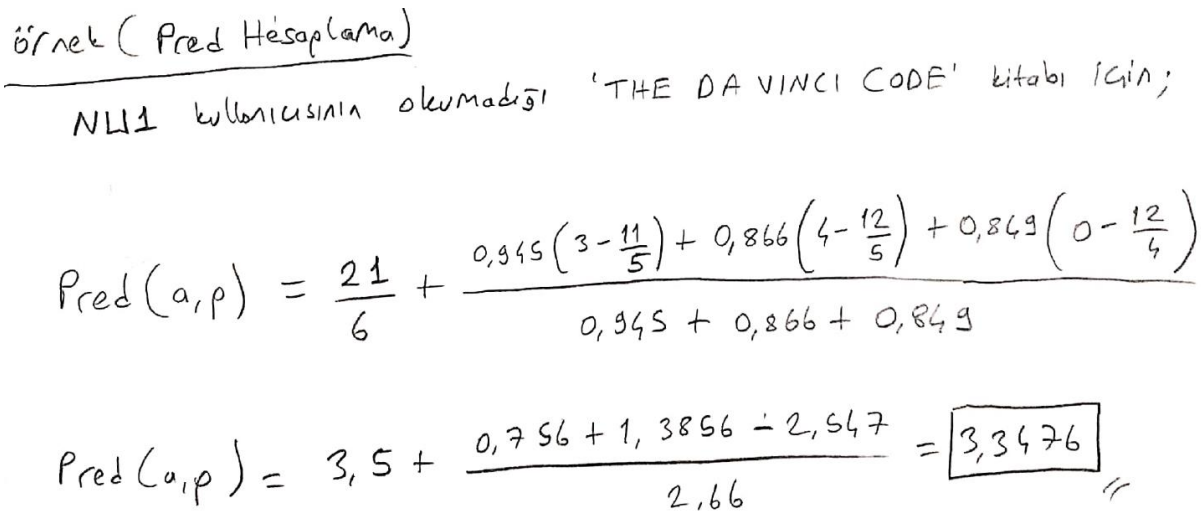
8-) int predictBook (calculatedSimilarities *sim, int k, char userName [70]) :

Bu fonksiyonda okuyucuya yeni kitap öneriyoruz. Parametre olarak gelen kullanıcı adını struct içinde var olup olmadığını döngü içinde koşul yaparak sorguluyoruz. Daha sonra parametre olarak gelen kullanıcının kitaplara verdiği puanların toplamının, okuduğu kitap sayısına bölünmesiyle ortalamayı elde ediyoruz. Ayrıca burada parametre olarak gelen kullanıcı adının okumadığı kitaplarında indisini tutuyoruz çünkü diğer kullanıcılar onun okumadığı kitap için kaç puan vermiş bu indis numaraları ile kontrol edeceğiz.

Formüle göre NU kullanıcısı için gereken değeri elde ettik. Şimdi sırada en benzer k kişinin bu kullanıcının okumadığı kitaplar için verdikleri puan üzerinden ilerleyeceğiz. Okunmayan kitap sayısı kadar bir döngü oluşturuyoruz ve daha sonra içinde k adet dönecek olan bir döngü

daha kuruyoruz çünkü ilk k kişiyi alacağız. Her U kullanıcısının NU'nun okumadığı kitap için verdiği puanı ve tüm kitaplara verdiği puanların toplamının, okuduğu kitaplara bölünmesiyle elde edilen değerleri işliyoruz formül için. Aynı zamanda her U kullanıcısının NU ile olan similarity değerlerinide formül içindeki işlemlere aktarıyoruz. Böylelikle her kitap için tahmini puan oluşturulacaktır. En yüksek puana sahip olan kitap ise kullanıcıya önerilecektir.

Örnek bir hesaplama :



9-) int main ( ) :

Programı kullanacak kişiden bir kullanıcı ismi girmesini istiyoruz. Daha sonra ise bir K değeri girmesini istiyoruz. Bu girdiler için gerekli fonksiyonlar çalışacak ve buna göre bir çıktı kullanıcıya sunulacaktır.

C KODU

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

typedef struct node {    // Dosyadan cekilmis verilerin tutuldugu struct
    char userName[50];       // Kullanici adi : U1...NU5
    char books[50][50];      // Kitaplar
    int givenPoint[8];       // Kitaplara verilen puanlar
    struct node *next;
}bookRecommendation;

bookRecommendation *front = NULL;
```

```
bookRecommendation *rear = NULL;

typedef struct nodeTwo{   // Hesaplanan similarity degerlerinin tutuldugu
struct yapisi
    char userName[50];      // Kullanici adi : NU1...NU5
    char who[50][50];       // Kullanici adi : U1...U20
    float similarity[20];   // Her NU1...NU5 kullanicilari icin U1....U20
arasindaki kullanicilar ile arasindaki similarity degerleri
    struct nodeTwo *next;
}calculatedSimilarities;

calculatedSimilarities *frontTwo = NULL;
calculatedSimilarities *rearTwo = NULL;

bookRecommendation *createNode(){     // Her kullanici ve onlarin okudugu
kitaplara verdigi puanlar icin bellekte yer aciyorum.
    bookRecommendation *newInformation =
(bookRecommendation*)calloc(sizeof(bookRecommendation),1);
    strcpy(newInformation->books[0],"TRUE BELIEVER");        // Kitap
isimleri ile islem yapmayacagim icin en basta direk buradan ekledim.Bunun
disinda hersey dosyadan cekiliyor.
    strcpy(newInformation->books[1],"THE DA VINCI CODE");
    strcpy(newInformation->books[2],"THE WORLD IS FLAT");
    strcpy(newInformation->books[3],"MY LIFE SO FAR");
    strcpy(newInformation->books[4],"THE TAKING");
    strcpy(newInformation->books[5],"THE KITE RUNNER");
    strcpy(newInformation->books[6],"RUNNY BABBIT");
    strcpy(newInformation->books[7],"HARRY POTTER");
    newInformation->next = NULL;
    return newInformation;
}

calculatedSimilarities *createNodeTwo(){    //  Similarity degerlerinin
tutuldugu struct yapisina gelecek olan her node icin burada yer aciyorum.
    calculatedSimilarities *newInformation =
(calculatedSimilarities*)calloc(sizeof(calculatedSimilarities),1);
    newInformation->next = NULL;
    return newInformation;
}

void enqueue(bookRecommendation *newInformation){   // Queue yapisi
kullandigim icin dosyadan cektigim her satiri kuyruga ekliyorum.
    if (front == NULL){        // Eger kuyruk bos ise front ve rear gelecek
olan ilk nodu gosterecek.
        front = newInformation;
        rear = newInformation;
    }
    else{                     // Onun haricinde eklenecek her node bir
sonraki siraya yerlesecek ve rear orayi gosterecek.
        rear->next = newInformation;
        rear = newInformation;
    }
}

void enqueue2(calculatedSimilarities *newInformation){      // Similarity
degerlerinin tutulduğu yerde de Queue yapisi kullandim.
    if (frontTwo == NULL){      // Eger kuyruk bos ise front ve rear
gelecek olan ilk nodu gosterecek.
        frontTwo = newInformation;
        rearTwo = newInformation;
    }
```

```
    else{                       // Onun haricinde eklenecek her node bir
sonraki siraya yerlesecek ve rear orayi gosterecek.
        rearTwo->next = newInformation;
        rearTwo = newInformation;
    }
}

void readTheFile(){    // Dosyadan veri okumak icin kullandigimiz fonksiyon
    FILE *file = fopen("RecomendationDataSet.csv","r");
    if(file == NULL){
        printf("Dosya okunamadi !");
    }
    else{
        printf("Dosya okuma basarili ! \n");
        char buff[1024];           // Dosyadaki verileri tutacağım dizi
        int rowCount = 0;          // Kitap isimlerinin olduğu satırı
atlayağım kontrol değişkeni
        int fieldCount;            //  Okuma yapacağım sütun için tuttuğum
sayaç
        int columnCount = 0;       // Kitap ismi icin kontrol degiskeni
        char bookNames[50][50];    // Kitap isimleri icin dizi
        char book[50];             // kitap isimleri gecici degisken
        bookRecommendation *newInformation;     // Struct yapısına aktarmak
üzere oluşturduğum geçiçi yapı
        while(fgets(buff,1024,file)){   // satır satır okuyorum dosyayı

            fieldCount = 0;
            rowCount++;
            int c=0;  // kitap isimleri icin sayac degiskeni
            int d=0;  //    kitap isimlerini alirken kullanilacak sayac
degiskeni
            int e=0;  //    kitap isimlerini alirken kullanilacak sayac
degiskeni
            if(rowCount == 1){     // Ilk satiri atliyor yani kitap
isimlerini almiyorum.
                while(buff[c] != '\0'){
                    if(buff[c] != ';' && buff[c] != '\n' && buff[c] !=
'\0'){
                        book[d] = buff[c];      // Her filmi book dizisine
karakter karakter dolduruyorum.
                        d++;
                    }
                    else{
                        columnCount++;
                        if(columnCount != 1){     // ilk satir ilk
sutundaki USERS metnini es geciyorum.
                            strcpy(bookNames[e],book);
                            e++;
                        }
                        int x;
                        for(x=0;x<sizeof(book)/sizeof(char);x++){   //
gecici diziyi temizliyorum.
                            book[x] = '\0';
                        }
                        book[d] = '\0';
                        d=0;
                    }
                    c++;
                }
                continue;       // ilk satiri aldiktan sonra artik bu if
yapisina girmeyecek.
```

```
            }
            int i=0;
            int j=0;
            int z;

            char field[50];   // Kullanıcı ismi ve verilen puanları
tutacağım değişken
            int b = -1;       // Verilen puanlar üzerinde dolaşabilmek için
sayaç değişkeni. -1'den başlama sebebi ilk başta Kullanıcı Adı geldiği için
orda da bir kere artıyo. Puana gelince sıfırdan başlıyor.
            newInformation = createNode();    // Bellekte yer acıyorum.

            while(buff[i] != '\0'){        // Okunan satirin sonuna kadar
karakter karakter bakiyorum
                if(buff[i] != ';' && buff[i] != '\n' && buff[i] != '\0'){
// Noktalı virgüllerle ayrılıyor bilgiler o yüzden
                    field[j] = buff[i];
// ';' gelene kadar karakterleri alıyoruz.
                    j++;
                }
                else{
                    int newField;

                    if(fieldCount == 0){        // İlk sütunda kullanıcı
adı var onu alıyoruz
                        strcpy(newInformation->userName,field);
                    }
                    if(fieldCount == 1){        // 1.Puan
                        if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){    // noktalı virgülden sonra yine noktalı virgül,null
ve bosluk gelirse
                            newField = 0;       // O halde o sütundaki
değer sıfır olacak.
                        }
                        else{
                            newField = atoi(field);         // Aksi halde
puanı int cast edip struct a aktarıyoruz.
                        }
                        newInformation->givenPoint[b] = newField;
                        strcpy(newInformation->books[b],bookNames[b]);
                    }
                    if(fieldCount == 2){        // 2.Puan
                        if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                            newField = 0;
                        }
                        else{
                            newField = atoi(field);
                        }
                        newInformation->givenPoint[b] = newField;
                        strcpy(newInformation->books[b],bookNames[b]);
                    }
                    if(fieldCount == 3){        // 3.Puan
                        if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                            newField = 0;
                        }
                        else{
                            newField = atoi(field);
                        }
                        newInformation->givenPoint[b] = newField;
```

```
                    strcpy(newInformation->books[b],bookNames[b]);
                }
                if(fieldCount == 4){        // 4.Puan
                    if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                        newField = 0;
                    }
                    else{
                        newField = atoi(field);
                    }
                    newInformation->givenPoint[b] = newField;
                    strcpy(newInformation->books[b],bookNames[b]);
                }
                if(fieldCount == 5){        // 5.puan
                    if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                        newField = 0;
                    }
                    else{
                        newField = atoi(field);
                    }
                    newInformation->givenPoint[b] = newField;
                    strcpy(newInformation->books[b],bookNames[b]);
                }
                if(fieldCount == 6){        // 6.Puan
                    if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                        newField = 0;
                    }
                    else{
                        newField = atoi(field);
                    }
                    newInformation->givenPoint[b] = newField;
                    strcpy(newInformation->books[b],bookNames[b]);
                }
                if(fieldCount == 7){        // 7.Puan
                    if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                        newField = 0;
                    }
                    else{
                        newField = atoi(field);
                    }
                    newInformation->givenPoint[b] = newField;
                    strcpy(newInformation->books[b],bookNames[b]);
                }
                if(fieldCount == 8){        //  8.Puan
                    if(buff[i+1] == ';' && buff[i+1] == '\n' &&
buff[i+1] == ' '){
                        newField = 0;
                    }
                    else{
                        newField = atoi(field);
                    }
                    newInformation->givenPoint[b] = newField;
                    strcpy(newInformation->books[b],bookNames[b]);
                }

                for(z=0;z<j;z++){    // gecici degisken olan field ı
temizliyoruz.
                    field[z] = '\0';
```

```
                    }
                    field[j] = '\0';
                    j=0;
                    fieldCount++;       // bir sonraki sütuna geçebilmek
için fieldCount u arttırıyoruz.
                    b++;
                }
                i++;
                field[j] = '\0';

            }
            i = 0;
            enqueue(newInformation);        // İlk satırı struct'a enqueue
ediyoruz.
        }
        fclose(file);
    }
}

void calculateSimilarity(char userName[70]){        // Similarity hesaplama
fonksiyonu
    if (front == NULL){      // Kuyruk boş ise Queue is clear mesajı
döndür.
        printf("\n Queue is clear");
        return;
    }
    bookRecommendation * temp = front;
    bookRecommendation * temp2 = front;
    while(temp!=NULL){     // Struct icerisinde NULL gelene kadar
dönüyoruz.
        if(strcmp(temp->userName,userName) == 0){    // Parametre olarak
gelen userName, struct icindeki userName ile eşleştiğinde işleme
başlıyoruz.
            calculatedSimilarities *toUsers;  // gecici toUsers
olustuyoruz.
            toUsers = createNodeTwo();     // 2.structta yer açıyoruz
gelecek olan similarity değerine.
            strcpy(toUsers->userName,temp->userName);
            int simCount = 0;
            while(strcmp(temp2->userName,"NU1") != 0){      // U1...U20
arasındaki kullanıcılar ile similarity değerleri oluşturuyoruz.
                int control = 0;    // kitaplara verilen puanları alabilmek
için indis değeri.
                int count=0;  // kitap sayısı kadar döngü kuracak değer.
                float averageU = 0;     // U Kullanıcısı için ortalama
değişkeni
                float averageN = 0;     // NU Kullanıcısı için ortalama
değişkeni
                int commonCount=0;      // ortak okunan kitap sayısı
                float totalU=0;         // U kullanıcısının verdiği toplam
puan
                float totalN=0;         // NU kullanıcısının verdiği toplam
puan
                int pointsU[sizeof(temp2->givenPoint) / sizeof(int)];   //
U1..U20 arası kullanıcıların, NU kullanıcısı ile ortak okuduğu kitaplara
verdiği puanların tutulduğu dizi
                int pointsN[sizeof(temp->givenPoint) / sizeof(int)];    //
NU1..NU5 arası kullanıcıların, U kullanıcısı ile ortak okuduğu kitaplara
verdiği puanların tutulduğu dizi

                while(count < sizeof(temp2->givenPoint) / sizeof(int)){
```

```
                    if(temp->givenPoint[control] != 0 && temp2-
>givenPoint[control] != 0 ){   // Her ikisinin verdiği puan sıfırdan farklı
ise ikiside o kitabı okumuştur.
                        totalU += temp2->givenPoint[control];       // U
kullanıcısının puanları toplanıyor.
                        totalN += temp->givenPoint[control];        // NU
kullanıcısının puanları toplanıyor.
                        pointsU[commonCount] = temp2->givenPoint[control];
// U kullanıcısının puanları diziye aktarılıyor.
                        pointsN[commonCount] = temp->givenPoint[control];
// NU kullanıcısının puanları diziye aktarılıyor.
                        commonCount++;
                    }
                    control++;
                    count++;
                }

                averageU = totalU / commonCount;     // U kullanıcısının
puanlarının ortalaması hesaplanıyor.
                averageN = totalN / commonCount;    // NU kullanıcısının
puanlarının ortalaması hesaplanıyor.
                int i=0;
                float pay = 0;
                float payda = 0;

                while(i<commonCount){
                    pay += (pointsN[i]-averageN) * (pointsU[i]-averageU);
// Formüldeki pay kısmı hesaplanıyor.
                    i++;
                }
                i=0;

                float x=0,y=0;
                while(i<commonCount){
                    x += (pointsN[i]-averageN)*(pointsN[i]-averageN);
                    y += (pointsU[i]-averageU)*(pointsU[i]-averageU);
                    i++;
                }
                payda = sqrt(x*y);   // formüldeki payda kısmı
hesaplanıyor.
                float result;
                result = pay / payda;  // sonuç bulunuyor.
                strcpy(toUsers->who[simCount],temp2->userName);
                toUsers->similarity[simCount] = result;
                simCount++;
                temp2 = temp2->next;
            }
            enqueue2(toUsers);    // similarity değerleri ile NU ve U
kullanıcılarıda enqueue ediliyor.
            temp2 = front;
        }
        temp = temp->next;
    }
}


calculatedSimilarities *findKUsers(char userName[50]){  // Bir okuyucunun
en benzer olduğu k kişinin belirlendiği fonksiyon
    calculatedSimilarities *temp = frontTwo;
    while(strcmp(temp->userName,userName) != 0){   // parametre olarak
gelen userName i struct içinde bulana kadar dönüyoruz.
```

```
        temp = temp->next;
    }
    if(strcmp(temp->userName,userName) == 0){
        printf("\n ******  %s icin similarity degerleri ****** \t\n", temp-
>userName);
        int i=0,j;
        float a;
        char user[50];
        while(i<sizeof(temp->similarity)/sizeof(float)){            //
Similarityleri sıraladık ve en büyükleri döndürüyoruz.
            for (j=i+1;j<sizeof(temp->similarity)/sizeof(float);++j)
            {
                if (temp->similarity[i] < temp->similarity[j])
                {
                    a =  temp->similarity[i];
                    strcpy(user,temp->who[i]);
                    temp->similarity[i] = temp->similarity[j];
                    strcpy(temp->who[i],temp->who[j]);
                    temp->similarity[j] = a;
                    strcpy(temp->who[j],user);
                }
            }
            i++;
        }
        return temp;
    }
}

int predictBook(calculatedSimilarities *sim,int k,char userName[50]){    //
Hangi kitap önerilecek bunun belirlendiği fonksiyon
    bookRecommendation *userN = front;
    while(strcmp(userN->userName,userName) != 0){   // parametre olarak
gelen userName i struct içinde bulana kadar dönüyoruz.
        userN = userN->next;
    }
    int i=0;
    float averageN = 0;    // NU Kullanıcısı için ortalama değişkeni
    float totalN=0;       // NU kullanıcısının verdiği toplam puan
    int notReadBooks = 0;  // NU kullanıcısının okumağı kitap sayısının
değişkeni
    int placedZero[sizeof(userN->givenPoint)/sizeof(int)];  // NU
kullanıcısında hangi indislerde 0 var yani okunmamış kitap hangi indiste
bunu tuttuk. U1..U20 kullanıcılarında işimize yarayacak.
    while(i<sizeof(userN->givenPoint)/sizeof(int)){
        if(userN->givenPoint[i] == 0){
            placedZero[notReadBooks] = i;   // Okumadığı kitapların
indislerini buluyoruz.
            notReadBooks++;                // Okumadigi kitap sayisi
        }
        totalN += userN->givenPoint[i];
        i++;
    }
    averageN = totalN / (sizeof(userN->givenPoint)/sizeof(int) -
notReadBooks);   // ra ortalama : 0 dan farkli olanlarin sayisina bolduk
yani okunmus kitap sayisini bulduk

    int a=0;

    printf("\n");

    bookRecommendation *userU = front;
```

```
    float pay=0;    // Formüldeki pay kısmı
    float payda = 0;  // Formüldeki payda kısmı
    float *result = (float*)calloc(sizeof(float),notReadBooks);  // Puan
tahmin değerlerinin tutulduğu dizi
    char bookNames[notReadBooks][150];  // Kitap isimleri


    i=0;

    while(i<notReadBooks){          // Okunmayan kitap sayısı kadar
dönüyoruz
        int j=0;
        while(j<k){                 // En benzer k kişi sayısı kadar
dönüyoruz.
            while(strcmp(userU->userName,"NU1") != 0){      //
Kullanıcıları U1...U20 arasında olacak şekilde bir döngü daha kuruyoruz.
                int count = 0;
                float totalU = 0;       // U kullanıcısının verdiği toplam
puan
                float averageU;         // U Kullanıcısı için ortalama
değişkeni
                int countZero = 0;      // U kullanıcısının okumadığı kitap
sayısı değişkeni
                while(count < sizeof(userU->givenPoint) / sizeof(int)){
// kitap sayısı kadar dönüp puanları inceliyoruz.
                    totalU += userU->givenPoint[count];
                    if(userU->givenPoint[count] == 0){
                        countZero++;
                    }
                    count++;

                }
                averageU = totalU / (sizeof(userU->givenPoint) /
sizeof(int) - countZero);    // rb ortalama : 0 dan farkli olanlarin
sayisina bolduk yani okunmus kitap sayisini bulduk
                if(strcmp(userU->userName,sim->who[j]) == 0){         //
Puanları alabilmek için ilk struct yapımıza eriştik kullanıcı ismi sorgusu
yaparak.
                    pay += sim->similarity[j] * (userU-
>givenPoint[placedZero[i]] - averageU);   // formüldeki pay hesaplanıyor..
                    payda += sim->similarity[j];        // formüldeki payda
hesaplanıyor..
                    strcpy(bookNames[i],userU->books[placedZero[i]]);
                }
                userU= userU->next;
            }
            userU = front;
            j++;
        }
        result[i] = averageN + (pay/payda);  // sonuç hesaplandı ve sonuç
dizisine aktarıldı.
        printf("\n'%s'---> %.3f",bookNames[i],result[i]);
        pay = 0;
        payda = 0;
        i++;
    }
    i=0;
    int j=0;
    float temporary;
    char bookName[70];
    while(i<notReadBooks){
```

```
        for (j=i+1;j<notReadBooks;++j){          // BURADA PUANI EN YÜKSEK
OLANI BELİRLİYORUZ ÇÜNKÜ ONU KULLANICIYA ÖNERECEĞİZ.
            if (result[i] < result[j])
            {
                temporary =  result[i];
                strcpy(bookName,bookNames[i]);
                result[i] = result[j];
                strcpy(bookNames[i],bookNames[j]);
                result[j] = temporary;
                strcpy(bookNames[j],bookName);
            }
        }
        i++;
    }
    printf("\n\n Onerilecek Kitap : %s",bookNames[0]);    // En yüksek
puanı olan kitap önerildi.
}


int main(){
    readTheFile();    // Dosya okuduk.
    char userName[70];      // Kullanıcının gireceği kullanıcı adı :
NU1...NU5
    printf("Kitap onerisi yapilacak kullanici adi : ");
    scanf("%s",userName);
    calculateSimilarity(userName);  // Similarity değerlerini
hesapladığımız fonksiyon
    int k;  // Bir okuyucunun en benzer olduğu k kişinin belirlenmesi
girilecek olan k değeri
    printf("\n\nEn benzer k kisi icin k degeri giriniz : ");
    scanf("%d",&k);

    calculatedSimilarities *temp = findKUsers(userName);    // Bir
okuyucunun en benzer olduğu k kişinin belirlendiği fonksiyon
    int i=0;
    while(i<k){
        printf("\n%s : %.3f",temp->who[i],temp->similarity[i]);    // En
benzer k kişiyi yazdırıyoruz similarity değerleri ile birlikte.
        i++;
    }

    predictBook(temp,k,userName);   // Son olarak hangi kitap önerilecek
bunun belirlendiği fonksiyon

    return 0;

}
```

EKRAN ÇIKTILARI

NU1 için :

```
Dosya okuma basarili !
Kitap onerisi yapilacak kullanici adi : NU1


En benzer k kisi icin k degeri giriniz : 3

 ******  NU1 icin similarity degerleri ******

U16 : 0.945
U5 : 0.866
U9 : 0.849

'THE DA VINCI CODE'---> 3.348
'RUNNY BABBIT'---> 2.726

 Onerilecek Kitap : THE DA VINCI CODE
--------------------------------
Process exited after 2.796 seconds with return value 0
Press any key to continue . . .
```

NU2 için :

```
Dosya okuma basarili !
Kitap onerisi yapilacak kullanici adi : NU2


En benzer k kisi icin k degeri giriniz : 6

 ******  NU2 icin similarity degerleri ******

U11 : 1.000
U2 : 0.982
U1 : 0.945
U19 : 0.866
U3 : 0.756
U12 : 0.756

'TRUE BELIEVER'---> 2.750
'THE KITE RUNNER'---> 2.231
'HARRY POTTER'---> 3.225

 Onerilecek Kitap : HARRY POTTER
--------------------------------
Process exited after 3.988 seconds with return value 0
Press any key to continue . . .
```

NU3 için :

```
Dosya okuma basarili !
Kitap onerisi yapilacak kullanici adi : NU3


En benzer k kisi icin k degeri giriniz : 5

 ******  NU3 icin similarity degerleri ******

U16 : 0.500
U14 : 0.498
U15 : 0.346
U6 : 0.114
U8 : -0.080

'THE WORLD IS FLAT'---> 0.725
'MY LIFE SO FAR'---> 0.972

 Onerilecek Kitap : MY LIFE SO FAR
--------------------------------
Process exited after 6.578 seconds with return value 0
Press any key to continue . . .
```

NU4 için :

```
Dosya okuma basarili !
Kitap onerisi yapilacak kullanici adi : NU4


En benzer k kisi icin k degeri giriniz : 6

 ******  NU4 icin similarity degerleri ******

U2 : 1.000
U13 : 1.000
U10 : 0.956
U4 : 0.866
U16 : 0.866
U3 : 0.845

'THE TAKING'---> 1.752
'RUNNY BABBIT'---> 1.620

 Onerilecek Kitap : THE TAKING
--------------------------------
Process exited after 8.426 seconds with return value 0
Press any key to continue . . .
```

NU5 için :

```
Dosya okuma basarili !
Kitap onerisi yapilacak kullanici adi : NU5


En benzer k kisi icin k degeri giriniz : 9

 ******  NU5 icin similarity degerleri ******

U9 : 0.982
U18 : 0.866
U7 : 0.853
U6 : 0.649
U16 : 0.642
U14 : 0.000
U2 : -0.500
U13 : -0.500
U17 : -0.693

'TRUE BELIEVER'---> 2.521
'THE KITE RUNNER'---> -2.109
'HARRY POTTER'---> 2.649

 Onerilecek Kitap : HARRY POTTER
--------------------------------
Process exited after 3.357 seconds with return value 0
Press any key to continue . . .
```