20.01.2021

YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM3021 ALGORİTMA ANALİZİ DÖNEM PROJESİ RAPORU

Kitap Öneri Sistemi

AHMET SAİD SAĞLAM

17011501

KONU

Prediction

Bu ödevde, işbirlikçi filtre (collaborative filtering) yöntemi ile bir kişinin önceki seçimlerine bakarak yeni kitap öneren bir sistem tasarlanması ve gerçeklenmesi istenmiştir.

ÇÖZÜM

Kütüphane Eklenmesi ve Makrolar

```c
/*
@file
BLM3021 2020-2021 GUZ Proje

İşbirlikçi filtre (collaborative filtering) yöntemi ile bir kişinin önceki seçimlerine bakarak
yeni kitap öneren bir sistem tasarımı.

@author
İsim: Ahmet Said SAĞLAM
Öğrenci No: 17011501
Tarih: 20.01.2021
E-Mail: l1117501@std.yildiz.edu.tr
Compiler: TDM-GCC 4.9.2 64 bit-Release
IDE: DEV-C++ (version 5.11)
İşletim Sistemi: Windows 10 Pro 64 bit
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <stdbool.h>
#include <ctype.h>
#include<math.h>
#define INPUT "recdataset.csv"
#define BUFFER_SIZE 1000 //txt dosyadan alınan satırın saklanacağı bufferin boyutu
#define WORD_SIZE 50
#define NAME_SIZE 20
#define AYRAC ";"   //kelimelerin ayrilacagi delim ifadesi
```

Kodun ilk kısmında gerekli olabilecek kütüphaneler eklenmiş, gerekli makrolar atanmış ve programın geliştiricisi ve geliştirilip çalıştırıldığı ortam hakkında bilgiler verilmiştir. Okuma yapılan dosyanın ismi INPUT isimli makroda belirtilmiştir.

findMax Fonksiyonu

```
39  //double arraydeki daha onceden maks olarak donmemis elemanlar arasındaki maks elemanın indisini donduren fonksiyon
40  int findMax(double *array, int *flags, int size) {
41      int i;
42      double temp;
43      int x = 0;
44
45
46      //maks hesabına dahil olmamıs ilk degerin indisini bul
47      while(flags[x] == 1) {
48          x++;
49      }
50
51      //son elemana gelindiyse kontrol etmeden don
52      if(x == (size - 1)) {
53          return x;
54      }
55
56      temp = array[x]; //degeri al
57
58      //diziyi x'ten itibaren gez
59      for(i = (x + 1); i < size; i++) {
60          //gezilen eleman tempten buyukse ve daha once donmediyse tempe al ve indisini sakla
61          if(temp < array[i] && flags[i] == 0) {
62              temp = array[i];
63              x = i;
64          }
65      }
66      flags[x] = 1;   //ilgili eleman donecek, flag degerini 1'e cek
67      return x;   //indisi dondur
68  }
```

findMax fonksiyonu flag dizisi yardımıyla daha önceden maksimum eleman olarak dönmemiş ve o anda dizinin maksimum elemanı olan elemanın indisini döndürür.

findMaxV2 Fonksiyonu

```
70  //double dizideki en buyuk elemanın indisini donduren fonksiyon
71  int findMaxV2(double *array, int size) {
72      int i;
73      double temp;
74      int x = 0;
75
76      temp = array[0];
77
78      for(i = 1; i < size; i++) {
79          if(temp < array[i]) {
80              temp = array[i];
81              x = i;
82          }
83      }
84      return x;
85  }
```

findMaxV2 fonksiyonu double dizinin maksimum elemanının indisini döndürür.

findSimilars Fonksiyonu

```
87  //icerisine aldıgı similarities dizininin maks k elamnının indislerini donduren fonksiyon
88  int *findSimilars(int size, double *similarities, int k) {
89      int i;
90      int *array = (int*) calloc(k, sizeof(int));
91      int *flags = (int*) calloc(size, sizeof(int));  //flags dizisine tum elemanları 0 olacak sekilde yer acılır
92
93      //en benzer k kisi bulunur
94      for(i = 0; i < k; i++) {
95          array[i] = findMax(similarities, flags, size);  //en benzer i. kisi diziye aktarılır
96      }
97
98      free(flags);
99
100     return array;   //dizi dondurulur
101 }
```

findSimilars fonksiyonu, seçilen yeni kullanıcıya en benzer k eski kullanıcının indislerini bir diziye aktarır ve döndürür.

allocateBookNames Fonksiyonu

```
103 //kitap isimlerini tutmak icin yer acan ve isimleri yerlestiren fonksiyon
104 char **allocateBookNames(char *buffer, int *book_count) {
105     char **bookNames;
106     char *token;
107     char *temp = (char*) calloc(BUFFER_SIZE, sizeof(char));
108     int count = 0;
109     int i;
110
111     //bufferin yedegi alınır
112     strcpy(temp,buffer);
113
114     token = strtok(buffer, AYRAC);
115     while(token != NULL) {
116         count++;    //kac adet kitap var sayilir
117         token = strtok(NULL, AYRAC);
118     }
119
120     count--; //ilk gozdeki gereksiz kelime counta sayılmaz
121
122     bookNames = (char**) calloc(count,sizeof(char*));   //kitap sayisi kadar, matriste satir icin yer acilir
123     for(i = 0; i < count; i++) {
124         bookNames[i] = (char*) calloc(WORD_SIZE, sizeof(char)); //maksimum kitap ismi uzunlugu WORD_SIZE kadar olacak sekilde matriste sutun yeri acilir
125     }
```

```
126
127     i = 0;  //matriste satir indisi
128
129     //kitap isimleri matrise kopyalanir
130     token = strtok(temp, AYRAC);    //ilk kelimeyi al isleme sokma
131     token = strtok(NULL, AYRAC);    //ikinci kelimeden itibaren kitap isimleri
132     while(token != NULL) {
133         strcpy(bookNames[i],token);
134         i++;
135         token = strtok(NULL, AYRAC);
136     }
137
138     //alınan son kitabın son harfi new line karakter ise temizlenir
139     if(bookNames[(count - 1)][strlen(bookNames[(count - 1)]) - 1] == '\n') {
140         bookNames[(count - 1)][strlen(bookNames[(count - 1)]) - 1] = '\0';
141     }
142
143     *book_count = count;    //kitap sayisi guncellenir
144
145     return bookNames;   //kitap matrisi dondurulur
146 }
```

allocateBookNames fonksiyonu içerisine aldığı bufferdan kitap sayısını ve isimlerini belirleyerek kitap sayısını bir değişkene, isimlerini ise bir matrise atayarak saklayan fonksiyondur.

userAllocator Fonksiyonu

```
148 //struct array olusturmak icin yer acan fonksiyon
149 struct user_info *userAllocator(int count) {
150     struct user_info *users;
151     users = (struct user_info*) calloc(count, sizeof(struct user_info));    //kullanici sayisi kadar struct yeri acilir
152
153     return users;   //struct dizisi disari dondurulur
154 }
```

userAllocator fonksiyonu içerisine aldığı sayı değeri kadar kullanıcı bilgilerini tutan dizi açan fonksiyondur.

fillStruct Fonksiyonu

```
156 //user struct dizisinin icini dolduran fonksiyon
157 struct user_info fillStruct(int *book_count, int old_user_count, char *buffer) {
158     int b_count = *book_count;  //sistemdeki kitap sayisi
159     int i;
160     char *token;
161     struct user_info temp_user; //gecici struct degiskeni
162
163     temp_user.points = (int*) calloc(b_count, sizeof(int)); //kitap sayisi kadar structun icinde points yeri acilir
164     temp_user.similarities = (double*) calloc(old_user_count, sizeof(double));  //eski kullanici sayisi kadar similarities dizisi icin yer acilir
165     temp_user.count = 0;    //kullanicin okudugu kitap sayisini tutan degisken 0'lanir
166
167     //user ismi temizlenir
168     for(i = 0; i < NAME_SIZE; i++) {
169         temp_user.name[i] = '\0';
170     }
171
172     token = strtok(buffer,AYRAC);   //kullanıcı ismini al
173     //printf("token isim : %s\n",token);
174     strcpy(temp_user.name, token);  //structa ata
175
176     //kullanicin kitaplar icin verdigi puanlar sirasiyla alinir ve points dizisine yerlestirilir
177     token = strtok(NULL,AYRAC);
178     for(i = 0; i < b_count; i++) {
179         temp_user.points[i] = atoi(token);
180         //printf("FILL KONTROL :string : %s, int : %d\n",token, atoi(token));
181         //kullanici kitabı okuduysa count arttirilir
182         if(temp_user.points[i] != 0) {
183             temp_user.count++;
184         }
185         token = strtok(NULL,AYRAC);
186     }
187     return temp_user;   //gecici kullanici disari dondurulur
188 }
```

fillStruct fonksiyonu kullanıcı dizisinin bir elemanı olarak atanacak structure için gerekli yerleri açar, bilgileri doldurur ve bu elemanı dışarı döndürür.

readFile Fonksiyonu

```c
int readFile(int *book_count, struct user_info **old_users_main, struct user_info **new_users_main, int *old_count, int *new_count, char ***book_names) {
    int old_user_count; //eski kullanıcı sayısını tutan degisken
    int new_user_count; //yeni kullanıcı sayısını tutan degisken
    FILE *inputFile; //file pointer
    char *buffer = (char*) calloc(BUFFER_SIZE,sizeof(char)); //dosyadan alinan satirin tutuldugu buffer
    char *temp = (char*) calloc(WORD_SIZE,sizeof(char));    //temporary dizi
    char *token;
    char **books;   //kitap isimlerini tutan matris
    int i, j;   //dongu degiskenleri
    struct user_info *old_users, *new_users;    //eski ve yeni kullanıcı bilgilerini tutan diziler

    //dosya acılamazsa hata verilir
    if((inputFile = fopen(INPUT,"r")) == NULL) {
        printf("Dosya okunmak icin acilamadi!\n");
        return 1;
    }
    //dosya acilirsa
    else {
        //dosyadaki ilk satır okunur
        fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile); //ilk satırı dosyadan buffer'a al
        //printf("BUFFER : %s\n",buffer);

        //eski kullanıcı sayısı okunur
        token = strtok(buffer,AYRAC);
        strcpy(temp,token);
        old_user_count = atoi(temp);
        old_users = userAllocator(old_user_count);  //eski kullanicilari tutmak icin dizi tanimlanir ve yer acilir
        *old_count = old_user_count;    //eski kullanici sayisi maindeki degiskenin icine aktarilir

        //yeni kullanıcı sayısı okunur
        token = strtok(NULL,AYRAC);
        strcpy(temp,token);
        new_user_count = atoi(temp);
        new_users = userAllocator(new_user_count);  //yeni kullanicilari tutmak icin dizi tanimlanir ve yer acilir
        *new_count = new_user_count;    //yeni kullanici sayisi maindeki degiskenin icine aktarilir

        //dosyadaki ikinci satır okunur
        fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile);
        //printf("BUFFER : %s\n",buffer);

        books = allocateBookNames(buffer, book_count);  //kitap isimleri alınır
        *book_names = books;    //maindeki matrise aktarılır

        //buffer temizlenir
        for(i = 0; i < BUFFER_SIZE; i++) {
            buffer[i] = '\0';
        }

        //dosyadaki satirlar eski kullanici sayisi kadar okunur ve eski kullanicilarin verileri struct dizisine kaydedilir
        for(i = 0; i < old_user_count; i++) {
            fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile); //satiri dosyadan oku
            old_users[i] = fillStruct(book_count,old_user_count,buffer);    //fonksiyondan donen kullaniciyi diziye ata
        }

        //dosyadaki satirlar yeni kullanici sayisi kadar okunur ve yeni kullanicilarin verileri struct dizisine kaydedilir
        for(i = 0; i < new_user_count; i++) {
            fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile); //satiri dosyadan oku
            new_users[i] = fillStruct(book_count,old_user_count,buffer);    //fonksiyondan donen kullaniciyi diziye ata
        }

        //eski ve yeni kullanıcı bilgileri maindeki dizilere aktarılır
        *old_users_main = old_users;
        *new_users_main = new_users;

        fclose(inputFile);
    }

    //free islemleri
    free(buffer);
    free(temp);

    return 0;
}
```

readFile fonksiyonu input dosyasını okur. Dosyadaki ilk satırda bulunan eski ve yeni kullanıcı sayılarını okuduktan sonra ikinci satırda bulunan kitap isimlerini ve bu isimleri sayarak kitap sayısını da okumuş olur. Son olarak ise ilk satırda okuduğu eski ve yeni kullanıcı sayılarına göre önce eski kullanıcı bilgileri sonra da yeni kullanıcı bilgileri olmak üzere dosyayı satır satır okuyarak tüm kullanıcıların bilgilerini sisteme kaydeder.

getAveragePoint Fonksiyonu

```
266 //içine aldığı kullanıcının okuduğu kitapların puan ortalamasını döndüren fonksiyon (estimationda kullanılan mean)
267 double getAveragePoint(struct user_info user, int book_count) {
268     int sum = 0;
269     int i;
270
271     for(i = 0; i < book_count; i++) {
272         sum += user.points[i];
273     }
274
275     return (double) sum / (double) user.count;
276 }
```

getAveragePoint fonksiyonu içerisine aldığı kullanıcının okuduğu kitapların puan ortalmasını döndürür.

getAveragePointV2 Fonksiyonu

```
278 //similarity hesabında kullanılan mean / içine aldığı iki kullanıcının ortak okudugu kitapların, 1. kullanıcı bakımından puan ortalamasını hesaplayan fonksiyon
279 double getAveragePointV2(struct user_info user_1, struct user_info user_2, int book_count) {
280     int sum = 0;
281     int i;
282     int common_count = 0;   //ortak kitap sayısı
283     for(i = 0; i < book_count; i++) {
284         //kitap ortak ise
285         if(user_1.points[i] != 0 && user_2.points[i] != 0) {
286             common_count++;
287             sum += user_1.points[i];
288         }
289     }
290     return (double) sum / (double) common_count;    //return mean
291 }
```

getAveragePointV2 fonksiyonu içerisine aldığı iki kullanıcının ortak okuduğu kitap sayısına göre ilk kullanıcının okuduğu kitapların puan ortalamasını döndürür.

getSimilarity Fonksiyonu

```c
//fonksiyon içerisine aldığı userların birbirlerine ne kadar benzediğini hesaplayıp döndürür
double getSimilarity(struct user_info user_1, struct user_info user_2, int book_count) {
    double similarity = 0.0;
    double parameter_1;
    double parameter_2;
    double avg_1;
    double avg_2;
    double sum_pay = 0.0;
    double sum_payda_1 = 0.0;
    double sum_payda_2 = 0.0;
    int i;

    //avg_1 = getAveragePoint(user_1, book_count);
    //avg_2 = getAveragePoint(user_2, book_count);

    //meanlar hesaplanır
    avg_1 = getAveragePointV2(user_1, user_2, book_count);
    avg_2 = getAveragePointV2(user_2, user_1, book_count);

    //iki user'ın da okudugu kitapları gez
    for(i = 0; i < book_count; i++) {
        //ilgili kitabı ikisi de okuduysa
        //pay ve paydaları hesaplayarak formulu uygula
        if(user_1.points[i] != 0 && user_2.points[i] != 0) {
            parameter_1 = (double) user_1.points[i] - avg_1;
            parameter_2 = (double) user_2.points[i] - avg_2;
            sum_pay += (parameter_1 * parameter_2);
            sum_payda_1 += pow(parameter_1, 2);
            sum_payda_2 += pow(parameter_2, 2);
        }
    }

    if(sum_payda_1 != 0 && sum_payda_2 != 0) {
        sum_payda_1 = sqrt(sum_payda_1);
        sum_payda_2 = sqrt(sum_payda_2);
        similarity = sum_pay / (sum_payda_1 * sum_payda_2);
    }

    return similarity;  //similartyi dondur
}
```

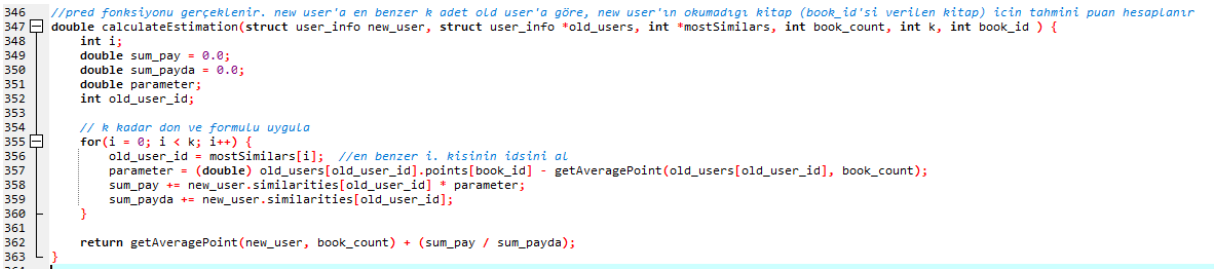
getSimilarity fonksiyonu içerisine aldığı iki kullanıcının birbirlerine olan benzerliklerini ödev dokümanında verilen fonksiyona göre hesaplayıp döndürür.

totalSimilarities Fonksiyonu

```c
//sistemdeki yeni kullanicilarin, tum eski kullanicilara benzerligini olcup kaydeden fonksiyon
void totalSimilarities(struct user_info *old_users, struct user_info *new_users, int old_user_count, int new_user_count, int book_count) {
    int i, j; //dongu degiskenleri

    for(i = 0; i < new_user_count; i++) {
        for(j = 0; j < old_user_count; j++) {
            //yeni kullanicilarin eski kullanicilara olan benzerlikleri hesaplanir
            new_users[i].similarities[j] = getSimilarity(new_users[i], old_users[j], book_count);
        }
    }
}
```

totalSimilarities fonksiyonu sistemdeki tüm yeni kullanıcıların, tüm eski kullanıcılara olan benzerliklerini ölçer ve kaydeder.

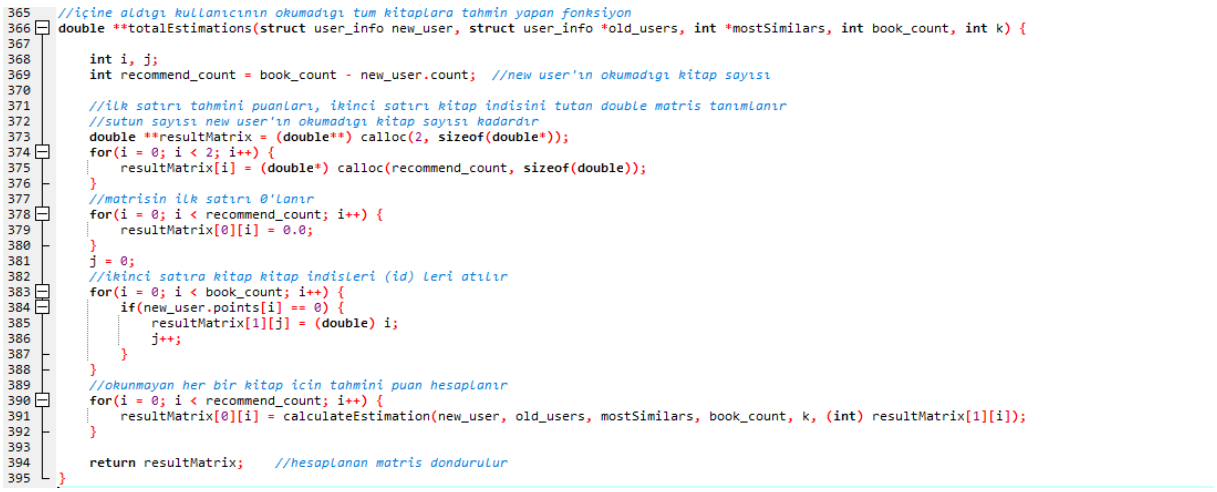
calculateEstimation Fonksiyonu



```c
//pred fonksiyonu gerçeklenir. new user'a en benzer k adet old user'a göre, new user'ın okumadığı kitap (book_id'si verilen kitap) için tahmini puan hesaplanır
double calculateEstimation(struct user_info new_user, struct user_info *old_users, int *mostSimilars, int book_count, int k, int book_id ) {
    int i;
    double sum_pay = 0.0;
    double sum_payda = 0.0;
    double parameter;
    int old_user_id;

    // k kadar dön ve formulu uygula
    for(i = 0; i < k; i++) {
        old_user_id = mostSimilars[i];  //en benzer i. kisinin idsini al
        parameter = (double) old_users[old_user_id].points[book_id] - getAveragePoint(old_users[old_user_id], book_count);
        sum_pay += new_user.similarities[old_user_id] * parameter;
        sum_payda += new_user.similarities[old_user_id];
    }

    return getAveragePoint(new_user, book_count) + (sum_pay / sum_payda);
}
```

calculateEstimation fonksiyonu içerisine aldığı yeni kullanıcı ve yeni kullanıcının idsi verilen okumadığı kitap için, eski kullanıcı verilerini kullanarak ödev dokümanının pred fonksiyonunda bahsedildiği üzere tahmini puan hesaplaması yapar ve hesaplanan puanı döndürür.

totalEstimations Fonksiyonu



```c
//içine aldıgı kullanıcının okumadıgı tum kitaplara tahmin yapan fonksiyon
double **totalEstimations(struct user_info new_user, struct user_info *old_users, int *mostSimilars, int book_count, int k) {

    int i, j;
    int recommend_count = book_count - new_user.count;  //new user'ın okumadıgı kitap sayısı

    //ilk satırı tahmini puanları, ikinci satırı kitap indisini tutan double matris tanımlanır
    //sutun sayısı new user'ın okumadıgı kitap sayısı kadardır
    double **resultMatrix = (double**) calloc(2, sizeof(double*));
    for(i = 0; i < 2; i++) {
        resultMatrix[i] = (double*) calloc(recommend_count, sizeof(double));
    }
    //matrisin ilk satırı 0'lanır
    for(i = 0; i < recommend_count; i++) {
        resultMatrix[0][i] = 0.0;
    }
    j = 0;
    //ikinci satıra kitap kitap indisleri (id) leri atılır
    for(i = 0; i < book_count; i++) {
        if(new_user.points[i] == 0) {
            resultMatrix[1][j] = (double) i;
            j++;
        }
    }
    //okunmayan her bir kitap için tahmini puan hesaplanır
    for(i = 0; i < recommend_count; i++) {
        resultMatrix[0][i] = calculateEstimation(new_user, old_users, mostSimilars, book_count, k, (int) resultMatrix[1][i]);
    }

    return resultMatrix;    //hesaplanan matris dondurulur
}
```

totalEstimations fonksiyonu içerisine aldığı yeni kullanıcın okumadığı tüm kitapların id ve tahmini puan değerlerini 2 satırlı matrisin gözlerine yerleştirir. Her bir sütunun ilk satırında tahmini puan değeri hesaplanarak yerleştirilirken 2. satırında ise o tahmini puanı hesaplanan kitabın idsi tutulur. Fonksiyon matrisi dışarı döndürür.

printTable ve printUser Fonksiyonları

```
397 //sistemdeki k degerini kullanicidan alip buna gore tahminleri tablo halinde her kullanici icin yazdiran fonksiyon
398 void printTable(int new_user_count, int old_user_count, struct user_info *new_users, struct user_info *old_users, int book_count, char **book_names) {
457
458 //sistemdeki k degerini ve kullanici isimini alip bu kullaniciya gore tahminleri yazdiran fonksiyon
459 void printUser(int new_user_count, int old_user_count, struct user_info *new_users, struct user_info *old_users, int book_count, char **book_names) {
533
```

Bu fonksiyonlar tüm hesaplamalar sonlandıktan sonra kullanıcıdan k değerini alır. printUser fonksiyonu ayrıca outputu özel olarak ekrana yazdırılmak istenen yeni kullanıcı ismini de kullanıcıdan alır. Hesaplanan değerlere ve alınan inputlara göre fonksiyonlar tahmini puan hesaplamalarını yapar ve ekrana bilgilendirme printleri atar.

main Fonksiyonu

```
534 int main() {
535
536     int book_count, old_user_count, new_user_count; //sistemdeki kitap ile birlikte eski ve yeni kullanici sayilarini tutan degiskenler
537     int choice; //kullanicinin veri goruntuleme secenegini tutan degisken
538     int cont = 1; //kullanıcı istedigi surece arama yapmasını saglayan while dongusune verilecek degisken
539     int file_control;   //dosyanın acilip acilamadigini kontrol eden degisken
540     char **book_names;  //kitap isimlerini tutuan matris
541     struct user_info *old_users, *new_users;    //eski ve yeni kullanici bilgilerini tutuan diziler
542
543
544     //veriler dosyadan okunur
545     file_control = readFile(&book_count, &old_users, &new_users, &old_user_count, &new_user_count, &book_names);
546     //dosya okunamadıysa program sonlandırılır
547     if(file_control == 1) {
548         return 0;
549     }
550
551     //yeni kullanicilarin eski kullanicilara olan benzerlikleri hesaplanir
552     totalSimilarities(old_users, new_users, old_user_count, new_user_count, book_count);
```

```
553
554     //kullanıcı istedigi surece donen while dongusu
555     while(cont) {
556
557         printf("Programi kullanici ozelinde calistirmak icin 1'e, tablo halinde tum kullanicilari gozlemlemek icin 0'a basiniz...\n");
558         scanf("%d",&choice);
559
560         //kullanıcın secimine gore output ekrana yazdırılır
561         if(choice) {
562             printUser(new_user_count, old_user_count, new_users, old_users, book_count, book_names);
563         }
564         else {
565             printTable(new_user_count, old_user_count, new_users, old_users, book_count, book_names);
566         }
567
568         printf("\nYeniden islem yapmak icin 1'e, cikmak icin 0'a basiniz.\n");
569         scanf("%d",&cont);
570         printf("\n");
571     }
572
573     return 0;
574 }
```

main fonksiyonunda öncelikle input dosyası okunur ve ardından benzerlik hesapları yapılır. Daha sonra kullanıcın isteğine göre veriler, her bir kullanıcı özelinde veya total olarak, kullanıcıdan alınan k değerine göre print fonksiyonlarıyla hesaplanır ve ekrana yazdırılır. Hesaplama ve ekrana bilgilendirme yazdırma işlemleri kullanıcı istediği sürece devam eder. En sonunda kullanıcının isteğine göre program sonlandırılıp kapanır.

Tahmini Puan Hesabında Algoritma Yaklaşımı

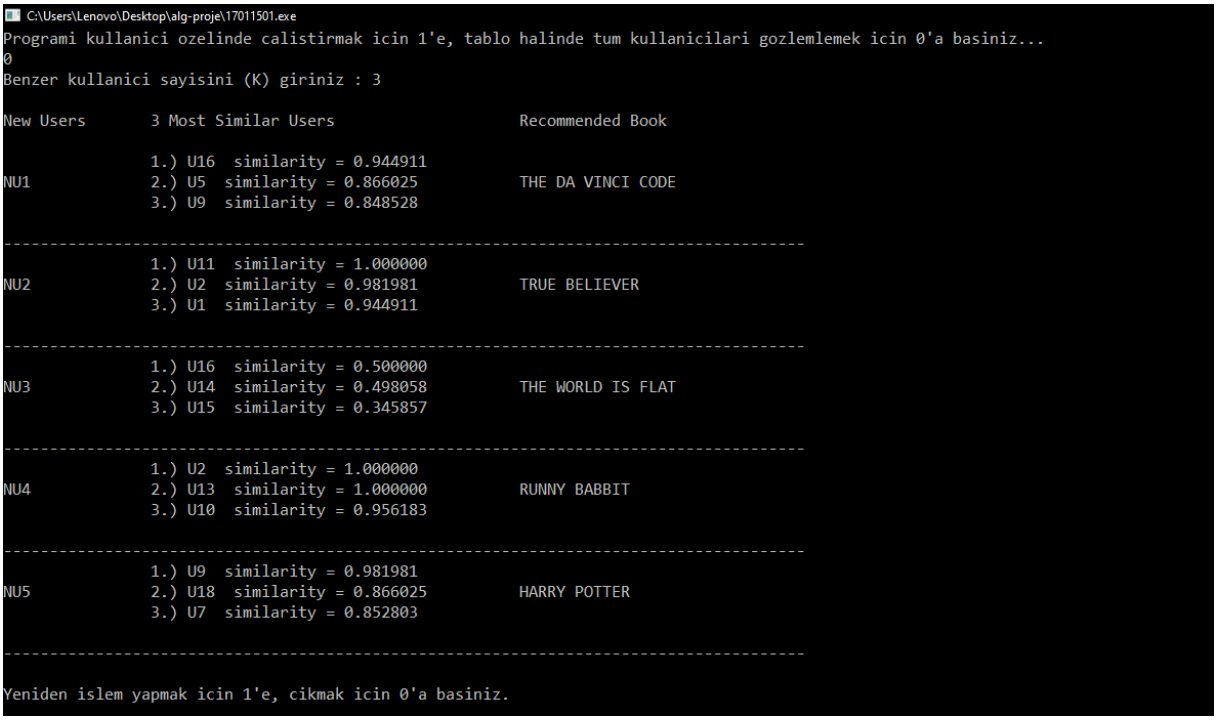
Tahmini puan hesabı alınan k değerine göre en benzer kullanıcılar maksimum similarity'e göre belirlenerek yapılmaktadır. Maksimum similarity her bir yeni kullanıcı için hesaplanan similarity dizisinin maksimum elemanını brute-force yöntemi ile bulmaktan geçer. Bir kez alınan similarity değeri flag dizisi yardımıyla bir sonraki aramalara dahil olmaz. Bu yöntemin karmaşıklığı brute-force olduğundan O(n)'dir. Ayrıca k kadar çağırıldığı için total karmaşıklık O(k*n) olur. k değerinin mantıken 1'e yakın olması bu karmaşıklığın O(n)' e yakınsamasını sağlamaktadır.

Okunan Input Dosyasının Formatı

	A	B	C	D	E	F	G	H	I	J	K	L
1	20	5										
2	USERS	TRUE BELI	THE DA VI	THE WORL	MY LIFE S	THE TAKIN	THE KITE R	RUNNY BA	HARRY POTTER			
3	U1	2	4	0	3	0	0	1	1			
4	U2	0	5	0	0	3	2	1	0			
5	U3	3	0	1	2	2	0	0	5			
6	U4	0	3	0	0	4	1	0	3			
7	U5	2	4	3	0	0	2	1	0			
8	U6	5	4	0	3	1	0	3	1			
9	U7	1	4	5	5	3	0	0	4			
10	U8	2	2	0	0	4	5	1	0			
11	U9	0	0	4	2	1	0	0	5			
12	U10	3	5	1	0	0	0	4	4			
13	U11	2	0	2	4	0	1	0	2			
14	U12	5	4	0	2	0	1	1	3			
15	U13	0	0	2	0	4	0	4	5			
16	U14	0	1	2	3	4	0	5	5			
17	U15	0	3	0	0	5	3	0	2			
18	U16	0	3	2	1	1	0	4	0			
19	U17	1	5	1	2	0	4	0	4			
20	U18	5	0	4	0	2	1	3	5			
21	U19	0	3	0	2	0	4	1	4			
22	U20	2	5	1	1	5	4	0	4			
23	NU1	4	0	5	3	2	3	0	4			
24	NU2	0	5	2	5	3	0	2	0			
25	NU3	2	1	0	0	2	3	4	1			
26	NU4	3	5	1	1	0	1	0	3			
27	NU5	0	2	3	2	1	0	3	0			

İlk satırda eski ve yeni kullanıcı değerleri sırasıyla verilmiştir. Kullanıcıların okumadığı kitapların puan değerleri 0 olarak belirlenmiştir. Input dosyasının formatında radikal bir değişiklik programın çalışmamasına sebebiyet verir!

PROGRAM ÇIKTILARI

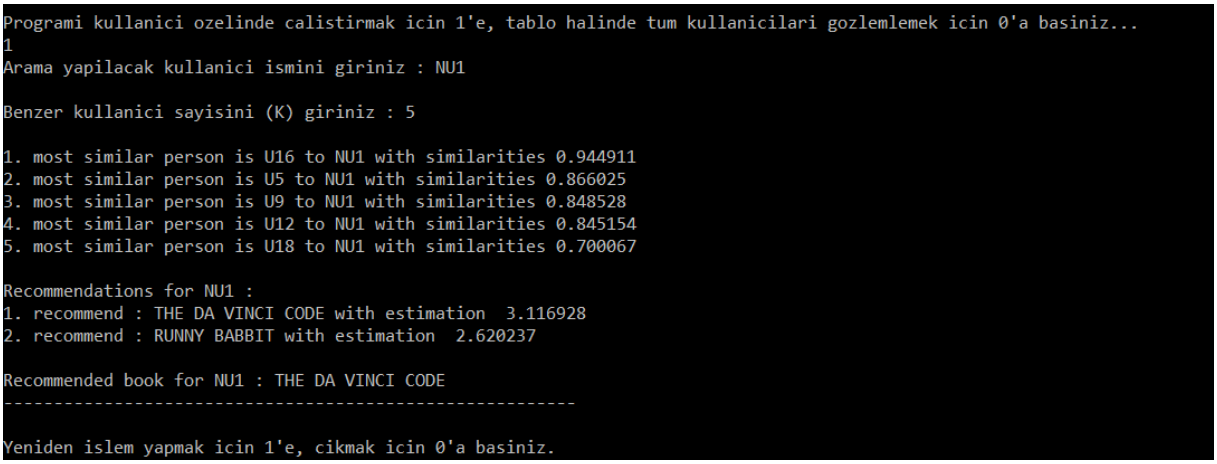


```
C:\Users\Lenovo\Desktop\alg-proje\17011501.exe
Programi kullanici ozelinde calistirmak icin 1'e, tablo halinde tum kullanicilari gozlemlemek icin 0'a basiniz...
0
Benzer kullanici sayisini (K) giriniz : 3

New Users       3 Most Similar Users                    Recommended Book

                1.) U16  similarity = 0.944911
NU1             2.) U5  similarity = 0.866025           THE DA VINCI CODE
                3.) U9  similarity = 0.848528

--------------------------------------------------------------------------------
                1.) U11  similarity = 1.000000
NU2             2.) U2  similarity = 0.981981           TRUE BELIEVER
                3.) U1  similarity = 0.944911

--------------------------------------------------------------------------------
                1.) U16  similarity = 0.500000
NU3             2.) U14  similarity = 0.498058          THE WORLD IS FLAT
                3.) U15  similarity = 0.345857

--------------------------------------------------------------------------------
                1.) U2  similarity = 1.000000
NU4             2.) U13  similarity = 1.000000          RUNNY BABBIT
                3.) U10  similarity = 0.956183

--------------------------------------------------------------------------------
                1.) U9  similarity = 0.981981
NU5             2.) U18  similarity = 0.866025          HARRY POTTER
                3.) U7  similarity = 0.852803

--------------------------------------------------------------------------------

Yeniden islem yapmak icin 1'e, cikmak icin 0'a basiniz.
```

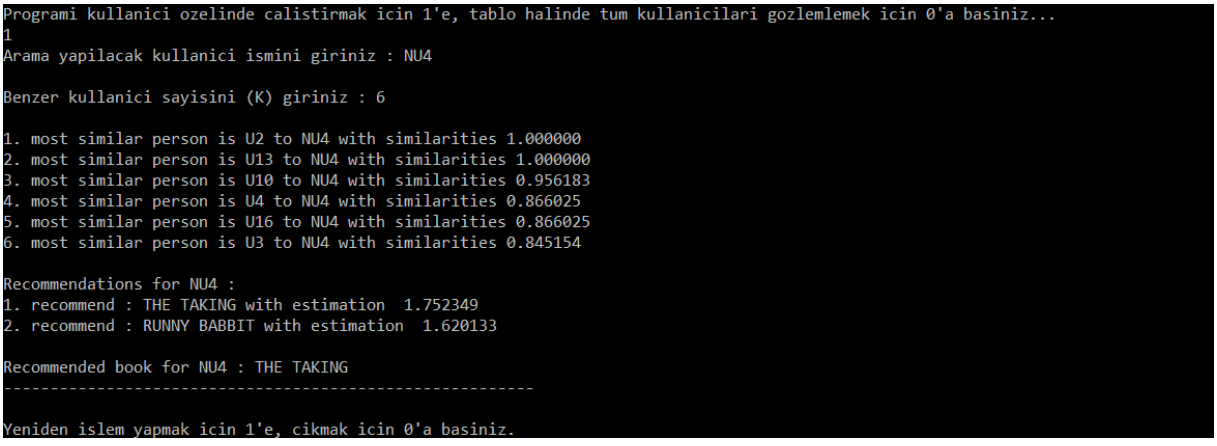
Tablo Halinde Output (k = 3)

```
Programi kullanici ozelinde calistirmak icin 1'e, tablo halinde tum kullanicilari gozlemlemek icin 0'a basiniz...
1
Arama yapilacak kullanici ismini giriniz : NU2

Benzer kullanici sayisini (K) giriniz : 4

1. most similar person is U11 to NU2 with similarities 1.000000
2. most similar person is U2 to NU2 with similarities 0.981981
3. most similar person is U1 to NU2 with similarities 0.944911
4. most similar person is U19 to NU2 with similarities 0.866025

Recommendations for NU2 :
1. recommend : TRUE BELIEVER with estimation  1.946159
2. recommend : THE KITE RUNNER with estimation  2.615364
3. recommend : HARRY POTTER with estimation  2.610341

Recommended book for NU2 : THE KITE RUNNER
--------------------------------------------------------

Yeniden islem yapmak icin 1'e, cikmak icin 0'a basiniz.
```

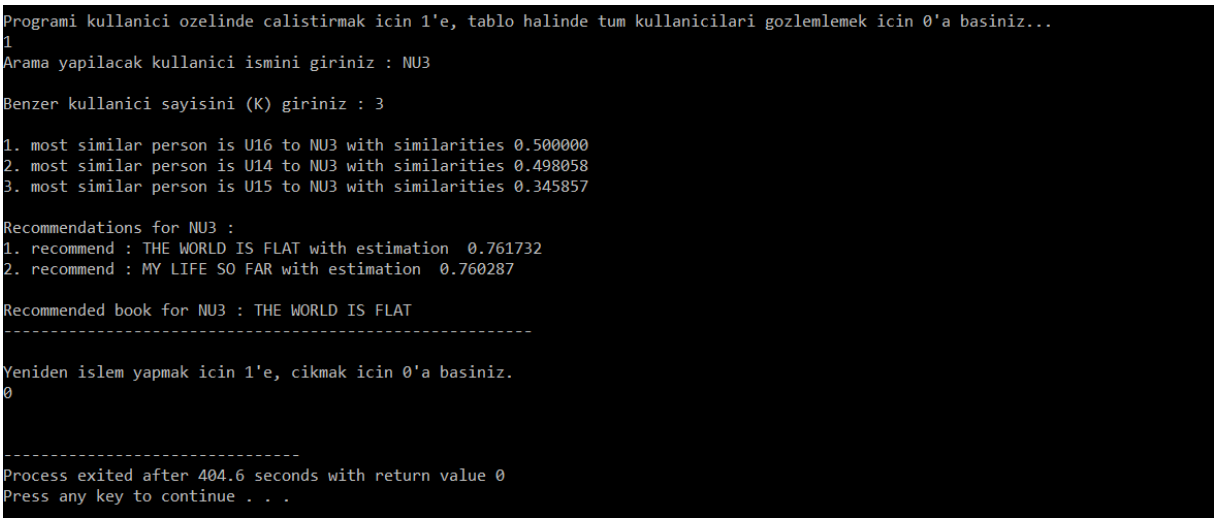
NU2 Kullanıcısı ve k = 4 Değeri İçin Output



NU1 Kullanıcısı ve k = 5 Değeri İçin Output



NU4 Kullanıcısı ve k = 6 Değeri İçin Output



NU3 Kullanıcısı ve k = 3 Değeri İçin Output

SOURCE CODE

```
1.  /*
2.  @file
3.  BLM3021 2020-2021 GUZ Proje
4.
5.  İşbirlikçi filtre (collaborative filtering) yöntemi ile bir kişinin önceki seçimleri
    ne bakarak
6.  yeni kitap öneren bir sistem tasarımı.
7.
8.  @author
9.  İsim: Ahmet Said SAĞLAM
10. Öğrenci No: 17011501
11. Tarih: 20.01.2021
12. E-Mail: l1117501@std.yildiz.edu.tr
13. Compiler: TDM-GCC 4.9.2 64 bit-Release
14. IDE: DEV-C++ (version 5.11)
15. İşletim Sistemi: Windows 10 Pro 64 bit
16. */
17.
18. #include <stdio.h>
19. #include <stdlib.h>
20. #include <string.h>
21. #include <conio.h>
22. #include <stdbool.h>
23. #include <ctype.h>
24. #include<math.h>
25. #define INPUT "RecomendationDataSet.csv"
26. #define BUFFER_SIZE 1000 //txt dosyadan alınan satırın saklanacağı bufferin boyutu
27. #define WORD_SIZE 50
28. #define NAME_SIZE 20
29. #define AYRAC ";"   //kelimelerin ayrilacagi delim ifadesi
30.
31. //her bir kullanıcı için kullanıcı bilgilerini tutan structure
32. typedef struct user_info {
33.     int *points;    //kitaplara verdigi puanları tutan dizi icin pointer
34.     char name[NAME_SIZE];   //kullanıcı ismi
35.     double *similarities;   //diger kullanıcılara benzerligini tutan dizi icin point
    er
36.     int count;  //kullanıcının okudugu kitap sayısı
37. }user_info;
38.
39. //double arraydeki daha onceden maks olarak donmemis elemanlar arasındaki maks elema
    nın indisini donduren fonksiyon
40. int findMax(double *array, int *flags, int size) {
41.     int i;
42.     double temp;
43.     int x = 0;
44.
45.
46.     //maks hesabına dahil olmamıs ilk degerin indisini bul
47.     while(flags[x] == 1) {
48.         x++;
49.     }
50.
51.     //son elemana gelindiyse kontrol etmeden don
52.     if(x == (size - 1)) {
53.         return x;
54.     }
55.
56.     temp = array[x]; //degeri al
57.
58.     //diziyi x'ten itibaren gez
```

```
59.     for(i = (x + 1); i < size; i++) {
60.         //gezilen eleman tempten buyukse ve daha once donmediyse tempe al ve indisin
    i sakla
61.         if(temp < array[i] && flags[i] == 0) {
62.             temp = array[i];
63.             x = i;
64.         }
65.     }
66.     flags[x] = 1;   //ilgili eleman donecek, flag degerini 1'e cek
67.     return x;   //indisi dondur
68. }
69. 
70. //double dizideki en buyuk elemanın indisini donduren fonksiyon
71. int findMaxV2(double *array, int size) {
72.     int i;
73.     double temp;
74.     int x = 0;
75. 
76.     temp = array[0];
77. 
78.     for(i = 1; i < size; i++) {
79.         if(temp < array[i]) {
80.             temp = array[i];
81.             x = i;
82.         }
83.     }
84.     return x;
85. }
86. 
87. //icerisine aldıgı similarities dizininin maks k elamnının indislerini donduren fonk
    siyon
88. int *findSimilars(int size, double *similarities, int k) {
89.     int i;
90.     int *array = (int*) calloc(k, sizeof(int));
91.     int *flags = (int*) calloc(size, sizeof(int));  //flags dizisine tum elemanları
    0 olacak sekilde yer acılır
92. 
93.     //en benzer k kisi bulunur
94.     for(i = 0; i < k; i++) {
95.         array[i] = findMax(similarities, flags, size);  //en benzer i. kisi diziye a
    ktarılır
96.     }
97. 
98.     free(flags);
99. 
100.            return array;   //dizi dondurulur
101.        }
102. 
103.       //kitap isimlerini tutmak icin yer acan ve isimleri yerlestiren fonksiyon
104.       char **allocateBookNames(char *buffer, int *book_count) {
105.           char **bookNames;
106.           char *token;
107.           char *temp = (char*) calloc(BUFFER_SIZE, sizeof(char));
108.           int count = 0;
109.           int i;
110. 
111.           //bufferin yedegi alınır
112.           strcpy(temp,buffer);
113. 
114.           token = strtok(buffer, AYRAC);
115.           while(token != NULL) {
116.               count++;    //kac adet kitap var sayilir
117.               token = strtok(NULL, AYRAC);
118.           }
119. 
120.           count--; //ilk gozdeki gereksiz kelime counta sayılmaz
```

```
121.
122.           bookNames = (char**) calloc(count,sizeof(char*));   //kitap sayisi kadar,
     matriste satir icin yer acilir
123.           for(i = 0; i < count; i++) {
124.               bookNames[i] = (char*) calloc(WORD_SIZE, sizeof(char)); //maksimum ki
    tap ismi uzunlugu WORD_SIZE kadar olacak sekilde matriste sutun yeri acilir
125.           }
126.
127.           i = 0;  //matriste satir indisi
128.
129.           //kitap isimleri matrise kopyalanir
130.           token = strtok(temp, AYRAC);    //ilk kelimeyi al isleme sokma
131.           token = strtok(NULL, AYRAC);    //ikinci kelimeden itibaren kitap isimler
    i
132.           while(token != NULL) {
133.               strcpy(bookNames[i],token);
134.               i++;
135.               token = strtok(NULL, AYRAC);
136.           }
137.
138.           //alınan son kitabın son harfi new line karakter ise temizlenir
139.           if(bookNames[(count - 1)][strlen(bookNames[(count - 1)]) -
     1] == '\n') {
140.               bookNames[(count - 1)][strlen(bookNames[(count - 1)]) - 1] = '\0';
141.           }
142.
143.           *book_count = count;    //kitap sayisi guncellenir
144.
145.           return bookNames;   //kitap matrisi dondurulur
146.       }
147.
148.       //struct array olusturmak icin yer acan fonksiyon
149.       struct user_info *userAllocator(int count) {
150.           struct user_info *users;
151.           users = (struct user_info*) calloc(count, sizeof(struct user_info));    /
    /kullanici sayisi kadar struct yeri acilir
152.
153.           return users;   //struct dizisi disari dondurulur
154.       }
155.
156.       //user struct dizisinin icini dolduran fonksiyon
157.       struct user_info fillStruct(int *book_count, int old_user_count, char *buffer
    ) {
158.           int b_count = *book_count;  //sistemdeki kitap sayisi
159.           int i;
160.           char *token;
161.           struct user_info temp_user; //gecici struct degiskeni
162.
163.           temp_user.points = (int*) calloc(b_count, sizeof(int)); //kitap sayisi ka
    dar structun icinde points yeri acilir
164.           temp_user.similarities = (double*) calloc(old_user_count, sizeof(double))
    ;  //eski kullanici sayisi kadar similarities dizisi icin yer acilir
165.           temp_user.count = 0;    //kullanicin okudugu kitap sayisini tutan degiske
    n 0'lanir
166.
167.           //user ismi temizlenir
168.           for(i = 0; i < NAME_SIZE; i++) {
169.               temp_user.name[i] = '\0';
170.           }
171.
172.           token = strtok(buffer,AYRAC);   //kullanıcı ismini al
173.           //printf("token isim : %s\n",token);
174.           strcpy(temp_user.name, token);  //structa ata
175.
176.           //kullanicin kitaplar icin verdigi puanlar sirasiyla alinir ve points diz
    isine yerlestirilir
```

```
177.            token = strtok(NULL,AYRAC);
178.            for(i = 0; i < b_count; i++) {
179.                temp_user.points[i] = atoi(token);
180.                //printf("FILL KONTROL :string : %s, int : %d\n",token, atoi(token));
181.                //kullanici kitabı okuduysa count arttirilir
182.                if(temp_user.points[i] != 0) {
183.                    temp_user.count++;
184.                }
185.                token = strtok(NULL,AYRAC);
186.            }
187.            return temp_user;   //gecici kullanici disari dondurulur
188.        }
189.
190.        //csv dosyasının ilk satırından eski ve yeni kullancı sayılarını, ikinci satırından kitap isimlerini ve diğer satırlarından kullanıcı verilerini okuyan fonksiyon
191.        int readFile(int *book_count, struct user_info **old_users_main, struct user_info **new_users_main, int *old_count, int *new_count, char ***book_names) {
192.            int old_user_count; //eski kullanıcı sayısını tutan degisken
193.            int new_user_count; //yeni kullanıcı sayısını tutan degisken
194.            FILE *inputFile; //file pointer
195.            char *buffer = (char*) calloc(BUFFER_SIZE,sizeof(char)); //dosyadan alinan satirin tutuldugu buffer
196.            char *temp = (char*) calloc(WORD_SIZE,sizeof(char));    //temporary dizi
197.            char *token;
198.            char **books;   //kitap isimlerini tutan matris
199.            int i, j;   //dongu degiskenleri
200.            struct user_info *old_users, *new_users;    //eski ve yeni kullanıcı bilgilerini tutan diziler
201.
202.            //dosya acılamazsa hata verilir
203.            if((inputFile = fopen(INPUT,"r")) == NULL) {
204.                printf("Dosya okunmak icin acilamadi!\n");
205.                return 1;
206.            }
207.            //dosya acilirsa
208.            else {
209.                //dosyadaki ilk satır okunur
210.                fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile); //ilk satırı dosyadan buffer'a al
211.                //printf("BUFFER : %s\n",buffer);
212.
213.                //eski kullanıcı sayısı okunur
214.                token = strtok(buffer,AYRAC);
215.                strcpy(temp,token);
216.                old_user_count = atoi(temp);
217.                old_users = userAllocator(old_user_count);  //eski kullanicilari tutmak icin dizi tanimlanir ve yer acilir
218.                *old_count = old_user_count;    //eski kullanici sayisi maindeki degiskenin icine aktarilir
219.
220.                //yeni kullanıcı sayısı okunur
221.                token = strtok(NULL,AYRAC);
222.                strcpy(temp,token);
223.                new_user_count = atoi(temp);
224.                new_users = userAllocator(new_user_count);  //yeni kullanicilari tutmak icin dizi tanimlanir ve yer acilir
225.                *new_count = new_user_count;    //yeni kullanici sayisi maindeki degiskenin icine aktarilir
226.
227.                //dosyadaki ikinci satır okunur
228.                fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile);
229.                //printf("BUFFER : %s\n",buffer);
230.
```

```
231.                books = allocateBookNames(buffer, book_count);  //kitap isimleri alın
    ır
232.                *book_names = books;    //maindeki matrise aktarılır
233.
234.                //buffer temizlenir
235.                for(i = 0; i < BUFFER_SIZE; i++) {
236.                    buffer[i] = '\0';
237.                }
238.
239.                //dosyadaki satirlar eski kullanici sayisi kadar okunur ve eski kulla
    nicilarin verileri struct dizisine kaydedilir
240.                for(i = 0; i < old_user_count; i++) {
241.                    fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile); //satiri dosy
    adan oku
242.                    old_users[i] = fillStruct(book_count,old_user_count,buffer);    /
    /fonksiyondan donen kullaniciyi diziye ata
243.                }
244.
245.                //dosyadaki satirlar yeni kullanici sayisi kadar okunur ve yeni kulla
    nicilarin verileri struct dizisine kaydedilir
246.                for(i = 0; i < new_user_count; i++) {
247.                    fgets(buffer,BUFFER_SIZE * sizeof(char),inputFile); //satiri dosy
    adan oku
248.                    new_users[i] = fillStruct(book_count,old_user_count,buffer);    /
    /fonksiyondan donen kullaniciyi diziye ata
249.                }
250.
251.                //eski ve yeni kullanıcı bilgileri maindeki dizilere aktarılır
252.                *old_users_main = old_users;
253.                *new_users_main = new_users;
254.
255.                fclose(inputFile);
256.            }
257.
258.            //free islemleri
259.            free(buffer);
260.            free(temp);
261.
262.            return 0;
263.
264.        }
265.
266.        //içine aldığı kullanıcının okuduğu kitapların puan ortalamasını döndüren fon
    ksiyon (estimationda kullanılan mean)
267.        double getAveragePoint(struct user_info user, int book_count) {
268.            int sum = 0;
269.            int i;
270.
271.            for(i = 0; i < book_count; i++) {
272.                sum += user.points[i];
273.            }
274.
275.            return (double) sum / (double) user.count;
276.        }
277.
278.        //similarity hesabında kullanılan mean / icine aldıgı iki kullanıcının ortak
    okudugu kitapların, 1. kullanıcı bakımından puan ortalamasını hesaplayan fonksiyon
279.        double getAveragePointV2(struct user_info user_1, struct user_info user_2, in
    t book_count) {
280.            int sum = 0;
281.            int i;
282.            int common_count = 0;   //ortak kitap sayısı
283.            for(i = 0; i < book_count; i++) {
284.                //kitap ortak ise
285.                if(user_1.points[i] != 0 && user_2.points[i] != 0) {
286.                    common_count++;
```

```
287.                     sum += user_1.points[i];
288.                 }
289.             }
290.             return (double) sum / (double) common_count;    //return mean
291.         }
292.
293.         //fonksiyon içerisine aldığı userların birbirlerine ne kadar benzediğini hesa
    playıp döndürür
294.         double getSimilarity(struct user_info user_1, struct user_info user_2, int bo
    ok_count) {
295.             double similarity = 0.0;
296.             double parameter_1;
297.             double parameter_2;
298.             double avg_1;
299.             double avg_2;
300.             double sum_pay = 0.0;
301.             double sum_payda_1 = 0.0;
302.             double sum_payda_2 = 0.0;
303.             int i;
304.
305.             //avg_1 = getAveragePoint(user_1, book_count);
306.             //avg_2 = getAveragePoint(user_2, book_count);
307.
308.             //meanlar hesaplanır
309.             avg_1 = getAveragePointV2(user_1, user_2, book_count);
310.             avg_2 = getAveragePointV2(user_2, user_1, book_count);
311.
312.             //iki user'ın da okudugu kitapları gez
313.             for(i = 0; i < book_count; i++) {
314.                 //ilgili kitabı ikisi de okuduysa
315.                 //pay ve paydaları hesaplayarak formulu uygula
316.                 if(user_1.points[i] != 0 && user_2.points[i] != 0) {
317.                     parameter_1 = (double) user_1.points[i] - avg_1;
318.                     parameter_2 = (double) user_2.points[i] - avg_2;
319.                     sum_pay += (parameter_1 * parameter_2);
320.                     sum_payda_1 += pow(parameter_1, 2);
321.                     sum_payda_2 += pow(parameter_2, 2);
322.                 }
323.             }
324.
325.             if(sum_payda_1 != 0 && sum_payda_2 != 0) {
326.                 sum_payda_1 = sqrt(sum_payda_1);
327.                 sum_payda_2 = sqrt(sum_payda_2);
328.                 similarity = sum_pay / (sum_payda_1 * sum_payda_2);
329.             }
330.
331.             return similarity;  //similartyi dondur
332.         }
333.
334.         //sistemdeki yeni kullanicilarin, tum eski kullanicilara benzerligini olcup k
    aydeden fonksiyon
335.         void totalSimilarities(struct user_info *old_users, struct user_info *new_use
    rs, int old_user_count, int new_user_count, int book_count) {
336.             int i, j; //dongu degiskenleri
337.
338.             for(i = 0; i < new_user_count; i++) {
339.                 for(j = 0; j < old_user_count; j++) {
340.                     //yeni kullanicilarin eski kullanicilara olan benzerlikleri hesap
    lanir
341.                     new_users[i].similarities[j] = getSimilarity(new_users[i], old_us
    ers[j], book_count);
342.                 }
343.             }
344.         }
345.
```

```
346.       //pred fonksiyonu gerçeklenir. new user'a en benzer k adet old user'a göre, n
   ew user'ın okumadığı kitap (book_id'si verilen kitap) icin tahmini puan hesaplanır
347.       double calculateEstimation(struct user_info new_user, struct user_info *old_u
   sers, int *mostSimilars, int book_count, int k, int book_id ) {
348.           int i;
349.           double sum_pay = 0.0;
350.           double sum_payda = 0.0;
351.           double parameter;
352.           int old_user_id;
353.
354.           // k kadar don ve formulu uygula
355.           for(i = 0; i < k; i++) {
356.               old_user_id = mostSimilars[i];  //en benzer i. kisinin idsini al
357.               parameter = (double) old_users[old_user_id].points[book_id] -
    getAveragePoint(old_users[old_user_id], book_count);
358.               sum_pay += new_user.similarities[old_user_id] * parameter;
359.               sum_payda += new_user.similarities[old_user_id];
360.           }
361.
362.           return getAveragePoint(new_user, book_count) + (sum_pay / sum_payda);
363.       }
364.
365.       //içine aldığı kullanıcının okumadığı tum kitaplara tahmin yapan fonksiyon
366.       double **totalEstimations(struct user_info new_user, struct user_info *old_us
   ers, int *mostSimilars, int book_count, int k) {
367.
368.           int i, j;
369.           int recommend_count = book_count -
    new_user.count;  //new user'ın okumadığı kitap sayısı
370.
371.           //ilk satırı tahmini puanları, ikinci satırı kitap indisini tutan double
   matris tanımlanır
372.           //sutun sayısı new user'ın okumadığı kitap sayısı kadardır
373.           double **resultMatrix = (double**) calloc(2, sizeof(double*));
374.           for(i = 0; i < 2; i++) {
375.               resultMatrix[i] = (double*) calloc(recommend_count, sizeof(double));

376.           }
377.           //matrisin ilk satırı 0'lanır
378.           for(i = 0; i < recommend_count; i++) {
379.               resultMatrix[0][i] = 0.0;
380.           }
381.           j = 0;
382.           //ikinci satıra kitap kitap indisleri (id) leri atılır
383.           for(i = 0; i < book_count; i++) {
384.               if(new_user.points[i] == 0) {
385.                   resultMatrix[1][j] = (double) i;
386.                   j++;
387.               }
388.           }
389.           //okunmayan her bir kitap icin tahmini puan hesaplanır
390.           for(i = 0; i < recommend_count; i++) {
391.               resultMatrix[0][i] = calculateEstimation(new_user, old_users, mostSim
   ilars, book_count, k, (int) resultMatrix[1][i]);
392.           }
393.
394.           return resultMatrix;    //hesaplanan matris dondurulur
395.       }
396.
397.       //sistemdeki k degerini kullanicidan alip buna gore tahminleri tablo halinde
   her kullanici icin yazdiran fonksiyon
398.       void printTable(int new_user_count, int old_user_count, struct user_info *new
   _users, struct user_info *old_users, int book_count, char **book_names) {
399.
400.           int i, j; //dongu degiskenleri
401.           int k; //k degeri (en benzer kac adet eski kullanici sayisi)
```

```
402.           int *similars; //en benzer k eski kullanicinin indisleri(id)
403.           int recommend_count; //kullanicinin okumadigi ve kullaniciya onerilecek o
    lan kitap sayisi
404.           int rec_id; //en fazla tahmini puana sahip okunmamis kitabın id'sini tuta
    n degisken
405.           double **estimate_results; //ilk satırı tahmini puanları, ikinci satırı o
    kunmamıs kitapların idlerini tutan matris
406.
407.
408.           //Verileri Tablo Seklinde yazdıran blok
409.           printf("Benzer kullanici sayisini (K) giriniz : ");
410.           scanf("%d",&k); //K degeri alinir
411.
412.           printf("\nNew Users\t%d Most Similar Users\t\t\tRecommended Book\n\n",k);

413.           for(i = 0; i < new_user_count; i++) {
414.               similars = findSimilars(old_user_count,new_users[i].similarities,k);

415.               recommend_count = book_count - new_users[i].count;
416.
417.               estimate_results = (double**) calloc(2, sizeof(double*));
418.               for(j = 0; j < 2; j++) {
419.                   estimate_results[j] = (double*) calloc(recommend_count, sizeof(do
    uble));
420.               }
421.
422.               estimate_results = totalEstimations(new_users[i],old_users, similars,
     book_count, k);   //tahmini puanlar hesaplanır
423.
424.
425.               //en fazla puana sahip okunmamıs kitap bulunur ve onerilir
426.               //okunmamıs kitap sayısı 1'den fazla ise
427.               if(recommend_count > 1) {
428.                   rec_id = findMaxV2(estimate_results[0], recommend_count);   //en
    fazla tahmini puana sahip okunmamıs kitabı bul
429.                   //printf("\nRecommended book for %s : %s\n",new_users[id].name, b
    ook_names[(int) estimate_results[1][rec_id]]); //en fazla puana sahip okunmamıs kita
    p onerilir
430.               }
431.               else {
432.                   rec_id = 0;
433.                   //printf("\nRecommended book for %s : %s\n",new_users[id].name, b
    ook_names[(int) estimate_results[1][0]]);  //en fazla puana sahip okunmamıs kitap on
    erilir
434.               }
435.
436.
437.               for(j = 0; j < k; j++) {
438.                   if(j == k / 2) {
439.                       printf("%s",new_users[i].name);
440.                       printf("\t\t%d.) %s  similarity = %lf",(j+1),old_users[simila
    rs[j]].name, new_users[i].similarities[similars[j]]);
441.                       printf("\t\t%s\n",book_names[(int) estimate_results[1][rec_id
    ]]);
442.                   }
443.                   else {
444.                       printf("\t\t%d.) %s  similarity = %lf\n",(j+1),old_users[simi
    lars[j]].name, new_users[i].similarities[similars[j]]);
445.                   }
446.               }
447.
448.               for(j = 0; j < recommend_count; j++) {
449.                   free(estimate_results[j]);
450.               }
451.               free(estimate_results);
452.               free(similars);
```

```
453.
454.               printf("\n-------------------------------------------------------------
    ---------------------------\n");
455.           }
456.       }
457.
458.       //sistemdeki k degerini ve kullanici isimini alip bu kullaniciya gore tahminl
    eri yazdiran fonksiyon
459.       void printUser(int new_user_count, int old_user_count, struct user_info *new_
    users, struct user_info *old_users, int book_count, char **book_names) {
460.
461.           int control_1 = 1; //dogru input(gecerli kullanici ismi) alimini kontrol
    eden degisken
462.           int i, j; //dongu degiskenleri
463.           int id; //oneri yapilacak new user'ın idsi
464.           int k; //k degeri (en benzer kac adet eski kullanici sayisi)
465.           int *similars; //en benzer k eski kullanicinin indisleri(id)
466.           int recommend_count; //kullanicinin okumadigi ve kullaniciya onerilecek o
    lan kitap sayisi
467.           int rec_id; //en fazla tahmini puana sahip okunmamis kitabın id'sini tuta
    n degisken
468.           char *user_name = (char*) calloc(NAME_SIZE, sizeof(char)); //oneri yapıla
    cak kullanici ismi
469.           double **estimate_results; //ilk satırı tahmini puanları, ikinci satırı o
    kunmamıs kitapların idlerini tutan matris
470.
471.           //inputu kontrol et, dogrusunu alana kadar don
472.           while(control_1) {
473.               id = 0;
474.               printf("Arama yapilacak kullanici ismini giriniz : ");
475.               scanf("%s",user_name);
476.               printf("\n");
477.               while(id < new_user_count && control_1 == 1) {
478.                   if(strcmp(new_users[id].name, user_name) == 0) {
479.                       control_1 = 0;
480.                   }
481.                   id++;
482.               }
483.               if(control_1) {
484.                   printf("Gecersiz kullanici ismi girdiniz! Tekrar giriniz.\n");
485.               }
486.           }
487.           id--;   //kullanici id'si belirlenir
488.           printf("Benzer kullanici sayisini (K) giriniz : ");
489.           scanf("%d",&k); //K degeri alinir
490.           printf("\n");
491.           similars = findSimilars(old_user_count,new_users[id].similarities,k);   /
    /id'si belli olan kisiye en benzer K kisinin id'si belirlenir
492.           for(i = 0; i < k; i++) {
493.               printf("%d. most similar person is %s to %s with similarities %lf\n",
    (i+1),old_users[similars[i]].name, new_users[id].name, new_users[id].similarities[si
    milars[i]]);
494.           }
495.
496.           recommend_count = book_count -
     new_users[id].count; //kullanıcının okumadıgı kitap sayısı belirlenir
497.
498.           //kullanıcının okumadıgı kitaplar icin tahmini puanları ve kitap id'lerin
    i tutan 2 satırlı matris icin yer acilir
499.           estimate_results = (double**) calloc(2, sizeof(double*));
500.           for(i = 0; i < 2; i++) {
501.               estimate_results[i] = (double*) calloc(recommend_count, sizeof(double
    ));
502.           }
503.
```

```
504.            estimate_results = totalEstimations(new_users[id],old_users, similars, bo
    ok_count, k);  //tahmini puanlar hesaplanır
505.
506.            //hesaplanan puanlara gore oneri kitapları ekrana yazdirilir
507.            printf("\nRecommendations for %s :\n",new_users[id].name);
508.            for(i = 0; i < recommend_count; i++) {
509.                printf("%d. recommend : %s with estimation % lf\n",(i+1), book_names[
    (int) estimate_results[1][i]], estimate_results[0][i]);
510.            }
511.
512.            //en fazla puana sahip okunmamıs kitap bulunur ve onerilir
513.            //okunmamıs kitap sayısı 1'den fazla ise
514.            if(recommend_count > 1) {
515.                rec_id = findMaxV2(estimate_results[0], recommend_count);   //en fazl
    a tahmini puana sahip okunmamıs kitabı bul
516.                printf("\nRecommended book for %s : %s\n",new_users[id].name, book_na
    mes[(int) estimate_results[1][rec_id]]);   //en fazla puana sahip okunmamıs kitap on
    erilir
517.            }
518.            else {
519.                printf("\nRecommended book for %s : %s\n",new_users[id].name, book_na
    mes[(int) estimate_results[1][0]]);    //en fazla puana sahip okunmamıs kitap oneril
    ir
520.            }
521.
522.            printf("------------------------------------------------------------\n");
523.
524.            //free islemleri
525.            for(i = 0; i < recommend_count; i++) {
526.                free(estimate_results[i]);
527.            }
528.            free(estimate_results);
529.            free(similars);
530.            free(user_name);
531.
532.        }
533.
534.        int main() {
535.
536.            int book_count, old_user_count, new_user_count; //sistemdeki kitap ile bi
    rlikte eski ve yeni kullanici sayilarini tutan degiskenler
537.            int choice; //kullanicinin veri goruntuleme secenegini tutan degisken
538.            int cont = 1; //kullanıcı istedigi surece arama yapmasını saglayan while
    dongusune verilecek degisken
539.            int file_control;   //dosyanın acilip acilamadigini kontrol eden degisken

540.            char **book_names;  //kitap isimlerini tutuan matris
541.            struct user_info *old_users, *new_users;    //eski ve yeni kullanici bilg
    ilerini tutuan diziler
542.
543.
544.            //veriler dosyadan okunur
545.            file_control = readFile(&book_count, &old_users, &new_users, &old_user_co
    unt, &new_user_count, &book_names);
546.            //dosya okunamadıysa program sonlandırılır
547.            if(file_control == 1) {
548.                return 0;
549.            }
550.
551.            //yeni kullanicilarin eski kullanicilara olan benzerlikleri hesaplanir
552.            totalSimilarities(old_users, new_users, old_user_count, new_user_count, b
    ook_count);
553.
554.            //kullanıcı istedigi surece donen while dongusu
555.            while(cont) {
556.
```

```
557.                printf("Programi kullanici ozelinde calistirmak icin 1'e, tablo halin
    de tum kullanicilari gozlemlemek icin 0'a basiniz...\n");
558.                scanf("%d",&choice);
559.
560.                //kullanıcın secimine gore output ekrana yazdırılır
561.                if(choice) {
562.                    printUser(new_user_count, old_user_count, new_users, old_users, b
    ook_count, book_names);
563.                }
564.                else {
565.                    printTable(new_user_count, old_user_count, new_users, old_users,
    book_count, book_names);
566.                }
567.
568.                printf("\nYeniden islem yapmak icin 1'e, cikmak icin 0'a basiniz.\n")
    ;
569.                scanf("%d",&cont);
570.                printf("\n");
571.            }
572.
573.            return 0;
574.        }
```