

YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



2020-2021 GÜZ YARIYILI
BLM3021 ALGORİTMA ANALİZİ DERSİ
PROJE RAPORU

Konu: Kitap Öneri Sistemi

Hazırlayan: Mehmet Hayri ÇAKIR – 16011023

Dersin Yürütücüsü: Dr. Öğr. Üyesi M. Amaç GÜVENSAN

İstanbul, Ocak 2021

İçindekiler Tablosu

1. Yöntem.....	2
1.1. Problem	2
1.2. Çözüm.....	2
2. Uygulama	2
2.1. Fonksiyonlar	2
2.1.1. char** populateBookNamesArray(FILE* fp, int* book_count).....	2
2.1.2. void insertUnreadBookRatings(char* line)	2
2.1.3. USER_DATA* populateUserArray(FILE* fp, int book_count, int* user_count, int* n_user_count)	2
2.1.4. void printBookNames(char** book_names, int book_count)	2
2.1.5. void printUserArray(USER_DATA* users, int user_count, int book_count).....	3
2.1.6. double similarity(USER_DATA* user_1, USER_DATA* user_2, int book_count)	3
2.1.7. int getUserIndex(USER_DATA* users, int user_count, char* user_name)	3
2.1.8. void twoUserSimilarity(USER_DATA* users, int user_count, int n_user_count, int book_count)	3
2.1.9. void sortSimilaritiesArray(SIMILARITY_DATA* similarities, int k)	3
2.1.10. void sortBookSuggestionArray(BOOK_SUGGESTION_DATA* suggestions, int unread_book_count).....	3
2.1.11. SIMILARITY_DATA* kMostSimilars(USER_DATA* users, int user_count, int n_user_count, int book_count, char* user_name, int user_index, int k).....	3
2.1.12. SIMILARITY_DATA* kMostSimilarsEntry(USER_DATA* users, int user_count, int n_user_count, int book_count)	3
2.1.13. void saveToFile(USER_DATA* users, SIMILARITY_DATA* similarities, char* user_name, char* most_suggested_book, int k)	3
2.1.14. void suggestBook(USER_DATA* users, char** book_names, int user_count, int n_user_count, int book_count)	3
2.1.15. int main()	3
2.2. Define Satırları ve Açıklamalar.....	4
2.3. Testler	4
2.4. Ekran Çıktıları.....	5
2.4.1. Program Ana Menüsü.....	5
2.4.2. İki Kullanıcı Benzerliğini Bulma İşlemi	5
2.4.3. Bir Kullanıcıya En Benzer k Adet Kullanıcının Bulunması İşlemi	7
2.4.4. Bir Kullanıcıya Kitap Önerilmesi Durumu	8
3. Kod	11

1. Yöntem

1.1. Problem

Günümüzde Youtube, Netflix, Amazon, Pinterest gibi internet ortamında milyonlarca kullanıcısı olan pek çok firma makine öğrenmesi tabanlı tavsiye sistemleri ile kullanıcılara kişiselleştirilmiş öneriler sunmaktadır. Bu projede de bahsedilen uygulamalara benzer bir sistem tasarlanması beklenmektedir.

1.2. Çözüm

Bu projede işbirlikçi filtre (collaborative filtering) yöntemi ile bir kişinin önceki seçimlerine bakarak yeni kitap öneren bir sistem tasarlanacak ve gerçekleştirilecektir.

2. Uygulama

2.1. Fonksiyonlar

Main fonksiyonu haricinde toplam 14 fonksiyon bulunmaktadır:

- `char** populateBookNamesArray(FILE* fp, int* book_count)`
- `void insertUnreadBookRatings(char* line)`
- `USER_DATA* populateUserArray(FILE* fp, int book_count, int* user_count, int* n_user_count)`
- `void printBookNames(char** book_names, int book_count)`
- `void printUserArray(USER_DATA* users, int user_count, int book_count)`
- `double similarity(USER_DATA* user_1, USER_DATA* user_2, int book_count)`
- `int getUserIndex(USER_DATA* users, int user_count, char* user_name)`
- `void twoUserSimilarity(USER_DATA* users, int user_count, int n_user_count, int book_count)`
- `void sortSimilaritiesArray(SIMILARITY_DATA* similarities, int k)`
- `void sortBookSuggestionArray(BOOK_SUGGESTION_DATA* suggestions, int unread_book_count)`
- `SIMILARITY_DATA* kMostSimilar(USER_DATA* users, int user_count, int n_user_count, int book_count, char* user_name, int user_index, int k)`
- `SIMILARITY_DATA* kMostSimilarEntry(USER_DATA* users, int user_count, int n_user_count, int book_count)`
- `void saveToFile(USER_DATA* users, SIMILARITY_DATA* similarities, char* user_name, char* most_suggested_book, int k)`
- `void suggestBook(USER_DATA* users, char** book_names, int user_count, int n_user_count, int book_count)`

2.1.1. `char** populateBookNamesArray(FILE* fp, int* book_count)`

Dosyadan ilk satırı okur ve kitap isimlerinin tutulduğu matrise dosyadaki kitap isimlerini yerleştirir. Parametre olarak gelen `book_count` pointer olarak gelir ve burada hesaplanır. Fonksiyondan kitap isimlerinin tutulduğu matris döndürülür.

2.1.2. `void insertUnreadBookRatings(char* line)`

Dosyada puan verilmeyen(okunmayan) kitapların puanını, 0 olarak güncelleyen fonksiyon. Strtok fonksiyonu token'ın null olduğu durumda çalışmadığı için strtok fonksiyonu öncesinde böyle bir işleme ihtiyaç duydum. Aynı zamanda kullanıcı, okunmayan kitaplar için 0, boşluk karakteri girerse veya hiçbir şey girmezse (null) de bu fonksiyon sayesinde hepsi 0 olarak güncelleniyor. Uniform bir görüntü oluşması sağlanıyor. Parametre olarak gelen pointer üzerinden düzenleme yapar.

2.1.3. `USER_DATA* populateUserArray(FILE* fp, int book_count, int* user_count, int* n_user_count)`

Dosyadaki her bir kullanıcı ve kitaplara verdiği puanlar bu fonksiyon ile okunur ve oluşturulan `USER_DATA` dizisi döndürülür.

2.1.4. `void printBookNames(char** book_names, int book_count)`

Kitap isimlerini yazdırmak için kullandığım bir fonksiyondu. Debug ederken kullanmak için yazdım fakat ihtiyaç olabilir diye düşünüp silmedim.

2.1.5. void printUserArray(USER_DATA* users, int user_count, int book_count)

Kullanıcıların isimleri ve kitaplara verdikleri puanları yazdırmak için kullandığım bir fonksiyondur. Debug ederken kullanmak için yazdım fakat ihtiyaç olabilir diye düşünüp silmedim.

2.1.6. double similarity(USER_DATA* user_1, USER_DATA* user_2, int book_count)

Parametre olarak gelen iki kullanıcının benzerliğini hesaplayıp döndüren fonksiyondur. Kullanıcıların kitaplara verdiği puanları hesaplarken ortak okunan kitaplara göre hesaplıyordum fakat bazı durumlarda 0/0 yani (nan) yani 0'a bölme hatası olduğu için, sonradan değiştirdim. Artık her kullanıcının ortalaması kendi okuduğu kitaplar üzerinden hesaplanıyor.

2.1.7. int getUserIndex(USER_DATA* users, int user_count, char* user_name)

Parametre olarak aldığı kullanıcı adını, kullanıcılar dizisinden bularak indisini döndüren fonksiyon.

2.1.8. void twoUserSimilarity(USER_DATA* users, int user_count, int n_user_count, int book_count)

Kullanıcı kitap önerisi yapmak yerine sadece iki kullanıcının benzerliğini hesaplamak isterse bu fonksiyon çağrılıp ilgili girdileri alır ve 2.1.5'teki asıl similarity fonksiyonunu çağırır.

2.1.9. void sortSimilaritiesArray(SIMILARITY_DATA* similarities, int k)

Ödev dokümanında, bütün kullanıcıları sıralamaktan daha efektif bir çözüm bulun dendiği için bu fonksiyonu yazdım. En benzer k kullanıcıyı bulurken, ilk k kullanıcı hesaplandıktan sonra benzerlikler dizisini küçükten büyüğe bu fonksiyon ile sıralıyorum. Böylece sonraki gelen kullanıcılarda, benzerlik ilk kullanıcıdan küçükse hiçbir işlem yapılmadan sonraki kullanıcıya devam edilecek, büyükse, ilgili indise yerleşmesi sağlanıp, o indisten öncekiler birer sola kaydırılacak ve ilk indisteki diziden silinmiş olacak. Yani kısaca, diziyi sonradan sıralamaktansa, her eleman geldiği anda ilgili indise yerleşiyor. Bütün kullanıcıları sıralamaya göre daha efektif bir çözüm.

2.1.10. void sortBookSuggestionArray(BOOK_SUGGESTION_DATA* suggestions, int unread_book_count)

Kitap önerileri oluşturulduktan sonra, ekrana yazdırırken güzel ve sıralı gözükmesi için diziyi sıralayan fonksiyondur.

2.1.11. SIMILARITY_DATA* kMostSimilars(USER_DATA* users, int user_count, int n_user_count, int book_count, char* user_name, int user_index, int k)

Parametre olarak adı ve bulunduğu indis verilen kullanıcıya en benzer k kullanıcının bulunduğu fonksiyondur. K uzunluğundaki benzerlikler ve kullanıcı indislerinin bulunduğu diziyi döndürür.

2.1.12. SIMILARITY_DATA* kMostSimilarsEntry(USER_DATA* users, int user_count, int n_user_count, int book_count)

Eğer kullanıcı kitap önerisi yapmak değil de sadece bir kullanıcıya en benzer k kullanıcıyı bulmak isterse bu fonksiyon çağrılır. Kullanıcıdan girdileri alır ve 2.1.11'deki kMostSimilars fonksiyonunu çağırır.

2.1.13. void saveToFile(USER_DATA* users, SIMILARITY_DATA* similarities, char* user_name, char* most_suggested_book, int k)

Kullanıcıya kitap önerisi yapıldıktan sonra, en benzer k kullanıcı ve önerilen kitap .csv uzantılı bir dosya oluşturulup ona yazılır. Dosya adı şöyle belirlenir:

- <kullanıcıadı>_kitap_önerisi_ve_en_benzer_<k>_kullanıcı.csv
Örneğin: NU1_kitap_önerisi_ve_en_benzer_3_kullanıcı.csv

2.1.14. void suggestBook(USER_DATA* users, char** book_names, int user_count, int n_user_count, int book_count)

Kullanıcıya kitap önerisi yapan fonksiyondur.

2.1.15. int main()

Kullanıcıdan girilen rakama göre ilgili fonksiyonu çağırır. Burada yapılabilecek 4 işlem vardır:

- 0 girilirse: program kapatılır.
- 1 girilirse: iki kullanıcının benzerliğini bulan fonksiyon çağrılır.
- 2 girilirse: adı verilen kullanıcıya en benzer k kullanıcıyı bulan fonksiyon çağrılır.
- 3 girilirse: adı verilen kullanıcıya kitap önerisi yapan fonksiyon çağrılır.

2.2. Define Satırları ve Açıklamalar

```
#define RECOMMENDATION_FILE_NAME "RecomendationDataSet.csv"
#define TOKEN_DELIMITERS ";\\n"
#define CSV_SEPERATOR ';'
#define BUFFER_SIZE 1024
#define MAX_WORD_SIZE 32
#define BOOK_SUGGESTIONS_DIRECTORY "Book Suggestions"
```

1) **RECOMMENDATION_FILE_NAME**

Kullanıcılar ve puanların tutulduğu dosya adını temsil eder. Farklı bir dosya ismi kullanılacağı zaman bu satır güncellenmelidir.

2) **TOKEN_DELIMITERS**

Strtok fonksiyonunda kullanılacak olan kelime ayırmaya yarayan karakterlerdir.

3) **CSV_SEPERATOR**

.csv uzantılı dosyalardan okuma yaparken, hücreleri ayıran karakterdir. Bazı bilgisayarlarda ',' (virgül) iken benim bilgisayarımda ';' (noktalı virgül) olduğu için bu şekilde tanımladım.

4) **BUFFER_SIZE**

Okunan satırların tutulacağı değişken boyutunu temsil eder. Bir satırdaki karakter sayısının bu değeri aşacağı düşünülüyorsa artırılmalıdır. Örneğin; kitap sayısı arttığında...

5) **MAX_WORD_SIZE**

Kullanıcı adı gibi çok fazla karakterden oluşmayan ifadelerin tutulacağı değişkenlerin boyutunu ifade ederken kullanılır. İhtiyaç duyulması durumunda artırılmalıdır.

6) **BOOK_SUGGESTIONS_DIRECTORY**

Her bir kullanıcı için kitap önerileri ve en benzer olduğu k kullanıcı bilgisi .csv uzantılı bir dosyaya yazıldığı senaryoda, dosya kalabalığı oluşmaması için bu isimde bir klasör, program ilk defa çalıştırıldığında oluşturulur. Sonrasında her oluşturulan .csv uzantılı dosya bu klasörün içine yazılır.

2.3. Testler

Ödev dokümanında bahsedilen testler için sonuçlar hem tablo hem de ekran çıktısı olarak, aşağıdadır. Aynı zamanda program kitap önerisi yaptıktan sonra önerilen kitap ve en benzer k kullanıcı bilgisini, her kullanıcı için ayrı dosyaya yazar.

Kullanıcı Adı	En Benzer 3 Kullanıcı			Önerilen Kitap Adı
NU1	U9	U12	U18	THE DA VINCI CODE
NU2	U2	U1	U11	TRUE BELIEVER
NU3	U16	U14	U15	THE WORLD IS FLAT
NU4	U2	U13	U16	RUNNY BABBIT
NU5	U9	U18	U6	HARRY POTTER

```
Enter user name to suggest a book: NU1
Enter k: 3
The most similar 3 users to 'NU1'
1| U9 | 0,848528
2| U12 | 0,843816
3| U18 | 0,700140
Book suggestions and predicted ratings for 'NU1'
1 | THE DA VINCI CODE | 1,930796
2 | RUNNY BABBIT | 1,750637
The book to be suggested is: 'THE DA VINCI CODE'
```

```

Enter user name to suggest a book: NU2
Enter k: 3
The most similar 3 users to 'NU2'
 1| U2 | 0,961210
 2| U1 | 0,951569
 3| U11 | 0,820693
Book suggestions and predicted ratings for 'NU2'
1 | TRUE BELIEVER | 2,303307
2 | THE KITE RUNNER | 2,010123
3 | HARRY POTTER | 1,955190
The book to be suggested is: 'TRUE BELIEVER'

```

```

Enter user name to suggest a book: NU3
Enter k: 3
The most similar 3 users to 'NU3'
 1| U16 | 0,510584
 2| U14 | 0,445087
 3| U15 | 0,309032
Book suggestions and predicted ratings for 'NU3'
1 | THE WORLD IS FLAT | 0,822540
2 | MY LIFE SO FAR | 0,770752
The book to be suggested is: 'THE WORLD IS FLAT'

```

```

Enter user name to suggest a book: NU4
Enter k: 3
The most similar 3 users to 'NU4'
 1| U2 | 0,989949
 2| U13 | 0,987763
 3| U16 | 0,841158
Book suggestions and predicted ratings for 'NU4'
1 | RUNNY BABBIT | 2,343484
2 | THE TAKING | 2,150650
The book to be suggested is: 'RUNNY BABBIT'

```

```

Enter user name to suggest a book: NU5
Enter k: 3
The most similar 3 users to 'NU5'
 1| U9 | 0,953313
 2| U18 | 0,740959
 3| U6 | 0,643325
Book suggestions and predicted ratings for 'NU5'
1 | HARRY POTTER | 3,039378
2 | TRUE BELIEVER | 2,101123
3 | THE KITE RUNNER | -0,542816
The book to be suggested is: 'HARRY POTTER'

```

2.4. Ekran Çıktıları

2.4.1. Program Ana Menüsü

```

0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: _

```

2.4.2. İki Kullanıcı Benzerliğini Bulma İşlemi

Ana menüdeyken 1 girilmesi durumunda, iki adet kullanıcı adı girdi olarak alınır ve bu iki kullanıcının benzerlikleri ekrana yazdırılır. Bununla ilgili farklı durumları gösteren 6 adet ekran çıktısı aşağıda verilmiştir.

1. U1, NU1 Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 1
Enter first user's name: U1
Enter second user's name: NU1
Similarity of 'U1' and 'NU1': -0,872357
```

2. U1, U8 Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 1
Enter first user's name: U1
Enter second user's name: U8
Similarity of 'U1' and 'U8': 0,190521
```

3. NU4, NU5 Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 1
Enter first user's name: NU4
Enter second user's name: NU5
Similarity of 'NU4' and 'NU5': -0,481125
```

4. Yanlış Kullanıcı Adı Girilmesi Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 1
Enter first user's name: asdf
Enter second user's name: NU1
Could not find user with name 'asdf'...Returning..
```

5. Aynı Kullanıcı Adının İki Kez Girilmesi Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 1
Enter first user's name: U1
Enter second user's name: U1
You entered same user name twice..Returning..
```

6. İki Kullanıcının Ortak Okuduğu Kitap Olmaması Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 1
Enter first user's name: U8
Enter second user's name: U9
'U8' and 'U9' has not have any common read books...Returning..
```

2.4.3. Bir Kullanıcıya En Benzer k Adet Kullanıcının Bulunması İşlemi

Ana menüdeyken 2 girilmesi durumunda, kullanıcı adı ve k sayısı girdi olarak alınıp, o kullanıcıya en benzer k kullanıcı ve benzerlik oranları bulunur. Bununla ilgili farklı durumları gösteren 5 adet ekran çıktısı aşağıda verilmiştir.

1. NU1, k = 5 Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 2
Enter user name: NU1
Enter k: 5
The most similar 5 users to 'NU1'
1| U9 | 0,848528
2| U12 | 0,843816
3| U18 | 0,700140
4| U5 | 0,658145
5| U16 | 0,563873
```

2. U1, k = 30 Durumu

U ile başlayan kullanıcı sayısı 20 olduğu için program ekrana bununla ilgili bir bilgi yazısı yazar ve k'yı 20'ye eşitler. Aynı zamanda U ile başlayan, U1 haricinde 19 kullanıcı olduğu için bu 19 kullanıcının benzerlik oranını yazdırır, U1'in kendisi ile olan benzerliği hesaplanmaz.

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 2
Enter user name: U1
Enter k: 30
You entered '30', but there are only '20' users..
The most similar 19 users to 'U1'
1| U2 | 0,997334
2| U5 | 0,987243
3| U11 | 0,648323
4| U6 | 0,473482
5| U15 | 0,380750
6| U12 | 0,354100
7| U10 | 0,330198
8| U17 | 0,298841
9| U7 | 0,257575
10| U4 | 0,196116
11| U8 | 0,190521
12| U20 | 0,112715
13| U19 | 0,073097
14| U16 | -0,315789
15| U18 | -0,475282
16| U13 | -0,832050
17| U3 | -0,942781
18| U14 | -0,976923
19| U9 | -0,992278
```


3. NU3, k = 22 Durumu

```
Enter the operation code: 2
Enter user name: NU3
Enter k: 22
You entered '22', but there are only '20' users..
The most similar 20 users to 'NU3'
 1| U16 | 0,510584
 2| U14 | 0,445087
 3| U15 | 0,309032
 4| U6  | 0,115919
 5| U7  | -0,047809
 6| U8  | -0,078047
 7| U10 | -0,305344
 8| U13 | -0,367926
 9| U11 | -0,412098
10| U17 | -0,428550
11| U20 | -0,443690
12| U1  | -0,467226
13| U18 | -0,542440
14| U4  | -0,556258
15| U9  | -0,600000
16| U19 | -0,608733
17| U12 | -0,723682
18| U5  | -0,915584
19| U3  | -0,927562
20| U2  | -0,941743
```

4. NU4, k= -2 Durumu

K pozitif girilmediği için program hata verir ve ana menüye döner.

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 2
Enter user name: NU4
Enter k: -2
k must be positive..Returning..
```

5. U1, k=1 Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 2
Enter user name: U1
Enter k: 1
The most similar 1 users to 'U1'
 1| U2 | 0,997334
```

2.4.4. Bir Kullanıcıya Kitap Önerilmesi Durumu

Ana menüdeyken 3 girilmesi durumunda kullanıcı adı ve k sayısı girdi olarak alınır, buna göre kullanıcıya bir kitap önerilir. Son olarak, en benzer k kullanıcı ve önerilen kitabın adı bir .csv uzantılı dosyaya yazılır. Aşağıda bununla ilgili farklı durumları gösteren 5 adet ekran çıktısı verilmiştir.

1. NU1, k=8 Durumu

Belirtilen durum için programın oluşturduğu çıktı ve .csv uzantılı dosyanın içeriği aşağıda verilmiştir.

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 3
Enter user name to suggest a book: NU1
Enter k: 8
The most similar 8 users to 'NU1'
 1| U9 | 0,848528
 2| U12 | 0,843816
 3| U18 | 0,700140
 4| U5 | 0,658145
 5| U16 | 0,563873
 6| U6 | 0,483548
 7| U19 | 0,246183
 8| U7 | 0,151994
Book suggestions and predicted ratings for 'NU1'
1 | THE DA VINCI CODE | 3,147232
2 | RUNNY BABBIT | 2,385375
The book to be suggested is: 'THE DA VINCI CODE'
```

	A	B	C	D	E	F	G	H	I
1	User name	NU1							
2	8 most similar users	U9	U12	U18	U5	U16	U6	U19	U7
3	Suggested book	THE DA VINCI CODE							

2. U7, k=5 Durumu

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 3
Enter user name to suggest a book: U7
Enter k: 5
The most similar 5 users to 'U7'
 1| U5 | 0,455842
 2| U11 | 0,432055
 3| U2 | 0,345705
 4| U9 | 0,311925
 5| U1 | 0,257575
Book suggestions and predicted ratings for 'U7'
1 | THE KITE RUNNER | 2,300953
2 | RUNNY BABBIT | 1,759649
The book to be suggested is: 'THE KITE RUNNER'
```

	A	B	C	D	E	F
1	User name	U7				
2	5 most similar users	U5	U11	U2	U9	U1
3	Suggested book	THE KITE RUNNER				

3. Kullanıcının Bütün Kitapları Okumuş Olması Durumu

Verilen dosyadaki U20 satırını güncelleyip bütün kitaplara puan vermesini sağladım. Yani U20'nin okumadığı kitap yok. Bu durum için oluşan ekran çıktısı aşağıdaki gibidir.

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 3
Enter user name to suggest a book: U20
Enter k: 5
'U20' has read all books...Returning..
```

4. Daha Önce Oluşturulan .csv Uzantılı Öneri Dosyasının Açık Olması Durumu

Bu durumda, işletim sisteminde bir dosya okuma modunda açık iken, program yazma modunda açmaya çalışacağı için hata oluşacaktır. Program bu hatayı da göz önünde bulundurmaktadır. "U7_kitap_önerisi_ve_en_benzer_5_kullanıcı.csv" isimli dosya açıkken, programa U7 ve k=5 verilmesi durumunda oluşan ekran çıktısı aşağıdadır.

```
0-Exit
1-Calculate two users' similarity
2-Calculate k most similar users to a user
3-Suggest new book to user

Enter the operation code: 3
Enter user name to suggest a book: U7
Enter k: 5
The most similar 5 users to 'U7'
 1| U5 | 0,455842
 2| U11 | 0,432055
 3| U2 | 0,345705
 4| U1 | 0,257575
 5| U8 | 0,233583
Book suggestions and predicted ratings for 'U7'
1 | THE KITE RUNNER | 3,079417
2 | RUNNY BABBIT | 1,971809
The book to be suggested is: 'THE KITE RUNNER'

Can't open output file for writing..
Make sure the following file is not open in your computer right now..Returning..
'U7_kitap_önerisi_ve_en_benzer_5_kullanıcı.csv'
```

5. NU4, k=9 Durumu

```
Enter user name to suggest a book: NU4
Enter k: 9
The most similar 9 users to 'NU4'
 1| U2 | 0,989949
 2| U13 | 0,987763
 3| U16 | 0,841158
 4| U10 | 0,817462
 5| U3 | 0,765784
 6| U12 | 0,764601
 7| U20 | 0,627473
 8| U18 | 0,623177
 9| U5 | 0,622222
Book suggestions and predicted ratings for 'NU4'
1 | RUNNY BABBIT | 1,934805
2 | THE TAKING | 1,325131
The book to be suggested is: 'RUNNY BABBIT'
```

	A	B	C	D	E	F	G	H	I	J
1	User name	NU4								
2	9 most similar users	U2	U13	U16	U10	U3	U12	U20	U18	U5
3	Suggested book	RUNNY BABBIT								

3. Kod

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <sys/stat.h>
#include <math.h>
#ifdef _WIN32
#include <windows.h>
#elif _WIN64
#include <windows.h>
#endif

#define RECOMMENDATION_FILE_NAME "RecomendationDataSet.csv"
#define TOKEN_DELIMITERS ";\n"
#define CSV_SEPERATOR ';' //benim bilgisayarimda csvden okuma yaparken ;(noktali virgul) karakteriyle ayrildigi icin olusturdum.
#define BUFFER_SIZE 1024 //kitap sayisinin artmasina bagli olarak bunun boyutunu artirmak gerekebilir. her bir satir icin max karakter sayisini temsil ediyor.
#define MAX_WORD_SIZE 32 //max kullanıcı adi uzunlugunu temsil ediyor.
#define BOOK_SUGGESTIONS_DIRECTORY "Book Suggestions" //kitap önerileri icin olusturulacak olan .csv uzantılı dosyaların konulacagi dosya adi, bir kereye mahsus olusturulur.

//Kullanici verisi tutulan veri yapisi
typedef struct
{
    char* user_name;
    int* book_ratings;
}USER_DATA;

//bir kullanıcıya en benzer k kullanıcı bulunurken bu veri yapısında tutuluyor.
typedef struct
{
    int user_index;
    double similarity;
}SIMILARITY_DATA;

//verilen kullanıcıya önerilen kitap indisi ve tahmin edilen puan bu veri yapısında tutuluyor.
typedef struct
{
    int book_index;
    double predicted_rating;
}BOOK_SUGGESTION_DATA;

//Kitaplari dosyadan okuyup, kitap isimleri matrisine yerlestiren fonksiyon.
char** populateBookNamesArray(FILE* fp, int* book_count)
{
    char** book_names; //kitap isimleri matrisi
    char line[BUFFER_SIZE]; //dosyadan okunan satir bu degiskene alinir
    char* token; //alınan stringi strtok ile tokenlara bolerken kullanılan degisken

    *book_count = 0;
    book_names = (char**)calloc(1, sizeof(char*));

    fgets(line, BUFFER_SIZE, fp);
    strtok(line, TOKEN_DELIMITERS);
```

```

//kitap sayisi kadar donen bu dongude, her bir kitap icin yer acilip string kopyalama islemi
yapiliyor. kitap sayisi degiskeni de guncelleniyor.
while ((token = strtok(NULL, TOKEN_DELIMITERS)) != NULL)
{
    (*book_count)++;
    book_names = (char**)realloc(book_names, (*book_count) * sizeof(char*));
    book_names[*book_count - 1] = (char*)calloc(strlen(token) + 1, sizeof(char));
    strcpy(book_names[*book_count - 1], token);
}
//son kitap icin olusan uc durum vardi. onu asmak icin boyle bir atama yapmak gerekiyor.
book_names[*book_count - 1][strlen(book_names[*book_count - 2])] = '\0';
free(token);
return book_names;
}

//dosyada puan verilmeyen(okunmayan) kitaplari, 0 puan olarak yerlestiren fonksiyon. strtok
fonksiyonu..
//null olan durumlarda calismadigi icin bu fonksiyonu yazdim.
void insertUnreadBookRatings(char* line)
{
    int i = 1; //dongu degiskeni
    int j = 0; //dongu degiskeni
    char new_line[BUFFER_SIZE]; //uzerinde string operasyonlari yapilacak gecici bi degisken
    for (i = 1; i < strlen(line); i++)
    {
        //eger mevcut karakter bi oncekiyle ayniyse
        if (line[i] == line[i - 1])
        {
            //eger bu karakter ayni zamanda csv separator karakterine esitse, burada bir
            okunmamis kitap vardir
            if (line[i] == CSV_SEPERATOR)
            {
                //iki csv separator arasina 0 puan oldugu bilgisi yerlestirilir.
                new_line[j] = CSV_SEPERATOR;
                j++;
                new_line[j] = '0';
                j++;
            }
            //yoksa devam ederiz.
            else
            {
                new_line[j] = line[i - 1];
                j++;
            }
        }
        //eger bosluk karakterine esitse, yine 0 puan yani okunmayan kitap vardir. 0
        yerlestiririz.
        else if (line[i - 1] == ' ')
        {
            new_line[j] = '0';
            j++;
        }
        //yoksa devam ederiz.
        else
        {
            new_line[j] = line[i - 1];
            j++;
        }
    }
    //en son kitap icin uc durum senaryosu mevcuttu. onu asmak icin de sondan bi onceki karakter
    csv separator ise son kitap okunmamistir.
    //0 yerlestiririz.
    if (line[i - 2] == CSV_SEPERATOR)
    {
        new_line[j] = '0';
        j++;
    }
}

```

```

}
//en sona da str terminator karakterini koyariz
new_line[j] = '\0';
//sonra parametre olarak gelen line degiskenini guncelleriz.
strcpy(line, new_line);
}

//dosyadan her bir kullanıcı ve kitaplara verdiği puanları okudugum fonksiyon.
USER_DATA* populateUserArray(FILE* fp, int book_count, int* user_count, int* n_user_count)
{
    USER_DATA* users; //kullanıcılar dizisi
    int eof_flag = 0; //dosya sonuna gelip gelmedigimi anlamak için kullanıyorum
    int i; //dongu degiskeni
    int temp_user_count; //gecici olarak kullandığım, kullanıcı sayısını tutan degisken.
    char* ch; //fgetsin donusunu bu degiskende sakliyorum
    char line[BUFFER_SIZE]; //okunan satiri bu degiskende sakliyorum
    char* token; //strtok fonksiyonu için kullandım
    users = (USER_DATA*)calloc(1, sizeof(USER_DATA));
    *user_count = 0;

    //dosya sonuna gelene kadar dongude kal.
    while (eof_flag != 1)
    {
        ch = fgets(line, BUFFER_SIZE, fp);
        //null keldiyse dosya sonudur.
        if (ch == NULL)
        {
            eof_flag = 1;
        }
        //eger asagidaki karakterlerden biri geldiyse, U ve NU satirlari arasindaki bos satir
        kisimindayiz demektir.
        //U kullanıcılarini ilgili degiskene aliriz.
        else if ((line[0] == ' ') || (line[0] == ';') || (line[0] == '\0') || (line == NULL))
        {
            temp_user_count = *user_count;
        }
        else
        {
            //okunmayan kitaplar için 0 puan yerlestirmesini yapariz.
            insertUnreadBookRatings(line);
            //kullanıcı adi alinir. kullanıcı adi bos degilse asagidaki islemler yapilir.
            if ((token = strtok(line, TOKEN_DELIMITERS)) != NULL)
            {
                //kullanıcı sayısı 1 artar.
                (*user_count)++;
                //ilgili yer acma islemleri yapilir.
                users = (USER_DATA*)realloc(users, (sizeof(USER_DATA) * (*user_count)));
                users[(*user_count) - 1].user_name = (char*)calloc(strlen(token),
                sizeof(char));
                strcpy(users[(*user_count) - 1].user_name, token);
                users[(*user_count) - 1].book_ratings = (int*)calloc(book_count,
                sizeof(int));
                //her bir kitap için
                for (i = 0; i < book_count; i++)
                {
                    //strtok ile verilen puan alinir.
                    token = strtok(NULL, TOKEN_DELIMITERS);
                    //csv dosyasinda okunmayan kitaplar için bazen 0 bazen null bazen
                    de bosluk karakteri geldigini gozlemledigim için,
                    //insertUnreadBookRatings fonksiyonuna ek olarak her ihtimale
                    karsi bu kontrolu de yapiyorum.
                    if ((token == NULL) || (strcmp(token, "") == 0) || (strcmp(token,
                    " ") == 0))
                    {
                        users[(*user_count) - 1].book_ratings[i] = 0;
                    }
                }
            }
        }
    }
}

```

```

        else
        {
            //tokeni integera donusturup ilgili atama yapilir.
            users[(*user_count) - 1].book_ratings[i] = atoi(token);
        }
    }
}

//n_user_count, NU seklinde baslayan kullanicilari temsil ediyor.
(*n_user_count) = (*user_count) - temp_user_count;
(*user_count) = temp_user_count;
free(ch);
return users;
}

//kitap adlarini yazdiran bir fonksiyon.
void printBookNames(char** book_names, int book_count)
{
    int i;
    for (i = 0; i < book_count; i++)
    {
        printf("-%s\n", book_names[i]);
    }
}

//kullanicilari yazdiran bir fonksiyon, kitaplara verdikleri oylarla birlikte.
void printUserArray(USER_DATA* users, int user_count, int book_count)
{
    int i;
    int j;
    for (i = 0; i < user_count; i++)
    {
        printf("%s\t", users[i].user_name);
        for (j = 0; j < book_count; j++)
        {
            printf("%d ", users[i].book_ratings[j]);
        }
        printf("\n");
    }
}

//iki kullanicinin benzerligini bulan fonksiyon.
double similarity(USER_DATA* user_1, USER_DATA* user_2, int book_count)
{
    int i; //dongu degiskeni
    int* common_read_book_indexes; //iki kullanicinin ortak okudugu kitaplarin, book_names
    matrisindeki satir indeksleri bu pointer dizisinde tutuluyor.
    int common_read_book_count = 0; //iki kullanicinin ortak okudugu kitap sayisi
    int user_1_read_book_count = 0; //1. kullanicinin okudugu kitap sayisi
    int user_2_read_book_count = 0; //2. kullanicinin okudugu kitap sayisi
    double sim = 0; //benzerlik degiskeni
    double sum = 0; //toplam icin kullanılan bir gecici degisken
    double sum2 = 0; //toplam icin kullanılan bir gecici degisken
    double mean_user_1 = 0; //1. kullanicinin puan ortalamasi
    double mean_user_2 = 0; //2. kullanicinin puan ortalamasi
    double current_sum_user_1 = 0; //toplam icin kullanılan bir degisken
    double current_sum_user_2 = 0; //toplam icin kullanılan bir degisken

    common_read_book_indexes = (int*)calloc(1, sizeof(int));
    //her bir kitap icin
    for (i = 0; i < book_count; i++)
    {
        //eger ikisi de 0 puan vermemisse ortak olarak okunmus bir kitaptir, ilgili atamalar
        yapilir.
        if ((user_1->book_ratings[i] != 0) && (user_2->book_ratings[i] != 0))

```

```

        {
            common_read_book_count++;
            common_read_book_indexes = (int*)realloc(common_read_book_indexes, sizeof(int) *
* common_read_book_count);
            common_read_book_indexes[common_read_book_count - 1] = i;
        }
    }
    //common_read_book_indexes = (int*)realloc(common_read_book_indexes, sizeof(int) *
common_read_book_count);
    //eger ortak okunmus bir kitap yoksa fonksiyondan doneriz.
    if (common_read_book_count == 0)
    {
        printf("'%' and '%' has not have any common read books...Returning..\n", user_1-
>user_name, user_2->user_name);
        return -2;
    }

    //ortalama hesaplarken ortak kitap sayisina gore hesapliyordum fakat bazi durumlarda
similarity= 0/0 yani (nan) ciktigini farketigim icin degistirdim.
    //her kullanicinin kendi okudugu kitap sayisina gore buluyorum.

    /*
    //her iki kullanicinin verdikleri puanlar toplanir.
    for (i = 0; i < common_read_book_count; i++)
    {
        mean_user_1 += user_1->book_ratings[common_read_book_indexes[i]];
        mean_user_2 += user_2->book_ratings[common_read_book_indexes[i]];
    }
    //ortak okunan kitap sayisina bolunerek puan ortalamalari bulunur.
    mean_user_1 /= common_read_book_count;
    mean_user_2 /= common_read_book_count;
    */
    //her iki kullanicinin verdikleri puanlar toplanir.
    //-----
    for (i = 0; i < book_count; i++)
    {
        if (user_1->book_ratings[i] != 0)
        {
            mean_user_1 += user_1->book_ratings[i];
            user_1_read_book_count++;
        }
        if (user_2->book_ratings[i] != 0)
        {
            mean_user_2 += user_2->book_ratings[i];
            user_2_read_book_count++;
        }
    }
    //okunan kitap sayisina bolunerek puan ortalamalari bulunur.
    mean_user_1 /= user_1_read_book_count;
    mean_user_2 /= user_2_read_book_count;
    //-----
    //bolme isleminin ust kismi hesaplanir.
    for (i = 0; i < common_read_book_count; i++)
    {
        sum = sum + (user_1->book_ratings[common_read_book_indexes[i]] - mean_user_1) *
(user_2->book_ratings[common_read_book_indexes[i]] - mean_user_2);
    }

    //bolme isleminin alt kismi hesaplanir.
    for (i = 0; i < common_read_book_count; i++)
    {
        current_sum_user_1 += pow((user_1->book_ratings[common_read_book_indexes[i]] -
mean_user_1), 2);
    }

```



```

        current_sum_user_2 += pow((user_2->book_ratings[common_read_book_indexes[i]] -
mean_user_2), 2);
    }
    //ilgili kok alma islemi yapilip sum2 hesaplanir.
    sum2 = sqrt(current_sum_user_1) * sqrt(current_sum_user_2);

    //son olarak bolme islemi yapilip benzerlik bulunur.
    sim = sum / sum2;
    free(common_read_book_indexes);
    return sim;
}

//kullanici adi verilen bir kullanicinin hangi indiste oldugu bilgisini donduren fonksiyon.
int getUserIndex(USER_DATA* users, int user_count, char* user_name)
{
    int i;
    for (i = 0; i < user_count; i++)
    {
        if (strcmp(user_name, users[i].user_name) == 0)
        {
            return i;
        }
    }
    printf("Could not find user with name '%s'...Returning..\n", user_name);
    return -1;
}

//similarity fonksiyonunun giris kismi, book suggestion haricinde sadece benzerlik bulunmak
istendigi durumlar icin,
//fonksiyonu iki parcaya boldum. book suggestion kullanilirsam inputlar book suggestion kisminde
alinir yoksa burada alinir.
void twoUserSimilarity(USER_DATA* users, int user_count, int n_user_count, int book_count)
{
    int user_1_index;
    int user_2_index;
    char user_1_name[MAX_WORD_SIZE];
    char user_2_name[MAX_WORD_SIZE];
    double sim;

    printf("Enter first user's name: ");
    scanf("%s", user_1_name);
    printf("Enter second user's name: ");
    scanf("%s", user_2_name);

    //girilen kullanıcı adlarına göre kullanıcı indisleri bulunur.
    user_1_index = getUserIndex(users, user_count + n_user_count, user_1_name);
    user_2_index = getUserIndex(users, user_count + n_user_count, user_2_name);

    if ((user_1_index == -1) || (user_2_index == -1))
    {
        return;
    }
    if (user_1_index == user_2_index)
    {
        printf("You entered same user name twice..Returning..\n");
        return;
    }
    sim = similarity(&users[user_1_index], &users[user_2_index], book_count);
    //eger sim = -2 ise, iki kullanıcının ortak okudugu kitap yoktur.
    if (sim == -2)
    {
        return;
    }
    printf("Similarity of '%s' and '%s': %f\n\n", user_1_name, user_2_name, sim);
}

```

//en benzer k kullaniciyi bulurken, ilk k kullanici hesaplandiktan sonra benzerlikler dizisini
kucukten buyuge siraladim.
//boylece sonraki gelen kullanicilarda, benzerlik ilk kullanicidan kucukse hic bir islem yapilmadan
devam edilecek, buyukse, ilgili indise yerlesmesi saglanip,
//o indisten oncekiler birer sola kaydirilacak ve ilk indisteki diziden silinmis olacak. Normal sort
islemine gore daha efektif bir cozum.

```
void sortSimilaritiesArray(SIMILARITY_DATA* similarities, int k)
{
    int i; //dongu degiskeni
    int j; //dongu degiskeni
    int x; //indis icin kullanılan bir degisken
    SIMILARITY_DATA temp;
    //k kez donulur.
    for (i = 0; i < k; i++)
    {
        //
        x = -1;
        //gecici benzerlik veri yapisinin benzerlik degiskeni 2 olarak ilkendirilir.
        (normalde max 1 oldugu icin ondan buyuk bir deger sectim.)
        temp.similarity = 2;
        //mevcut indisten dizinin sonuna kadar gidilir ve siralama islemi yapilir.
        for (j = i; j < k; j++)
        {
            if (similarities[j].similarity < temp.similarity)
            {
                temp = similarities[j];
                x = j;
            }
        }
        if (x != -1)
        {
            similarities[x] = similarities[i];
            similarities[i] = temp;
        }
    }
}
```

//ciktida guzel gorunmesi icin kitap onerilerini tahmini puanlar buyukten kucuge olacak sekilde
siralayan fonksiyon.

```
void sortBookSuggestionArray(BOOK_SUGGESTION_DATA* suggestions, int unread_book_count)
{
    int i; //dongu degiskeni
    int j; //dongu degiskeni
    int x; //indis erisimi icin kullanılan bir degisken
    BOOK_SUGGESTION_DATA temp;

    //okunmayan kitapların siralama islemi yapilir.
    for (i = 0; i < unread_book_count; i++)
    {
        x = -1;
        temp.predicted_rating = -1;
        for (j = i; j < unread_book_count; j++)
        {
            if (suggestions[j].predicted_rating > temp.predicted_rating)
            {
                temp = suggestions[j];
                x = j;
            }
        }
        if (x != -1)
        {
            suggestions[x] = suggestions[i];
            suggestions[i] = temp;
        }
    }
}
```

```

//en benzer k kullanıcının bulunduğu fonksiyon.
SIMILARITY_DATA* kMostSimilar(USER_DATA* users, int user_count, int n_user_count, int book_count,
char* user_name, int user_index, int k)
{
    int i; //dongu degiskeni
    int j; //dongu degiskeni
    int x; //dongu degiskeni
    int same_index_flag = 0; //kullanıcının kendisiyle benzerliğinin bulunmasının önüne geçmek
    için kullandığım flag
    SIMILARITY_DATA* similarities; //benzerlikler dizisi.
    double sim; //benzerlik degiskeni

    //benzerlikler dizisine k kadar yer acıyorum
    similarities = (SIMILARITY_DATA*)calloc(k, sizeof(SIMILARITY_DATA));
    //her bir kullanıcı için
    for (i = 0; i < k; i++)
    {
        //eger k benzer kullanıcı bulunacak olan kullanıcı, ilk k elemanın içindeyse
        kendisiyle benzerliği bulunmasın diye kontrol ediyorum.
        if (i != user_index)
        {
            //kullanıcı indisi atıyorum ve benzerliği similarity fonksiyonuyla bulup onu da
            atıyorum.
            similarities[i].user_index = i;
            similarities[i].similarity = similarity(&users[user_index], &users[i],
            book_count);
        }
        else
        {
            //benzerliği bulunmak istenenle mevcut indis aynıysa flagi set ediyorum.
            same_index_flag = 1;
        }
    }
    //eger for döngüsünden çıktığımda same_index_flag set edilmişse, ilgili yere döngüden
    çıktığım andaki i ve similarityyi atıyorum, yani k+1. kullanıcının oluyorum.
    if (same_index_flag)
    {
        similarities[user_index].user_index = i;
        similarities[user_index].similarity = similarity(&users[user_index], &users[i],
        book_count);
    }

    //kullanıcılar dizisindeki ilk k kullanıcıya baktıktan sonra k uzunluğundaki benzerlikler
    dizisini küçükten büyüğe sıralıyorum.
    sortSimilaritiesArray(similarities, k);

    //bundan sonra ise, eğer gelen kullanıcının benzerliği, benzerlikler dizisi[0] dan küçükse,
    zaten o diziyeye giremez diyip geçiyorum.
    //büyükse de küçük olduğu kısma gelene kadar dizi üzerinde ilerleyip ilgili gozu bulunca
    insert ediyorum. benzerlikler dizisi[0] in yerinde [1], [1] in yerine [2] gelecek şekilde,,
    //kullanıcının yerleştiği gözden geriye doğru bir adım kaydırma işlemi yapıyor. gelen
    kullanıcının direkt ilgili goze yerleşmesi, bütün kullanıcıları sıralamaktan daha efektif bir
    yöntem.

    //eger same_index_flag set ise, k+1 yoksa k'dan başlayıp kullanıcı sayısına kadar döngüde
    kalınır.
    for (i = (same_index_flag ? k + 1 : k); i < user_count; i++)
    {
        //eger i, kullanıcı indisine eşitse işlem yapmayız.
        if (i == user_index)
        {
            continue;
        }
    }
}

```

```

        j = 0;
        //kullanici ve i. kullanici arasindaki benzerlik hesaplanir.
        sim = similarity(&users[user_index], &users[i], book_count);
        //hesaplanan benzerlige gore i. kullanicinin hangi indise yerlesecegi hesaplanir.
        while ((j < k) && (sim > similarities[j].similarity))
        {
            j++;
        }
        //j artmamis i. kullanicinin benzerligi, similarities dizisindeki butun
        benzerliklerden kucuktur, devam ederiz.
        if (j == 0)
        {
            continue;
        }
        //yoksa ilgili indisten oncesini birer asagi kaydirir ve 0. indistekini yok ederiz.
        for (x = 0; x < j - 1; x++)
        {
            similarities[x].similarity = similarities[x + 1].similarity;
            similarities[x].user_index = similarities[x + 1].user_index;
        }
        //sonra i. kisinin benzerligi ve indisi atanir diziye.
        similarities[j - 1].similarity = sim;
        similarities[j - 1].user_index = i;
    }

    //yazdirma islemleri.
    printf("The most similar %d users to '%s'\n", k, user_name);
    for (i = k - 1; i > -1; i--)
    {
        printf("%3d| %-3s | %10f\n", k - i, users[similarities[i].user_index].user_name,
        similarities[i].similarity);
    }

    return similarities;
}

```

//en benzer k kisiyi buldugumuz fonksiyonun girisi. book suggestion cagrilirsa input alinma islemi orada, yoksa sadece k benzer bulunmak istenirse input burada alinabilmesi icin,
//iki parcaya boldum fonksiyonu.

```

SIMILARITY_DATA* kMostSimilarEntry(USER_DATA* users, int user_count, int n_user_count, int
book_count)
{

```

```

    char user_name[MAX_WORD_SIZE];
    int user_index;
    int k;
    printf("Enter user name: ");
    scanf("%s", user_name);
    user_index = getUserIndex(users, user_count + n_user_count, user_name);
    if (user_index == -1)
    {
        return NULL;
    }
    printf("Enter k: ");
    scanf("%d", &k);
    //k negatif girilmesi durumunu kontrol ederiz.
    if (k < 1)
    {
        printf("k must be positive..Returning..\n");
        return NULL;
    }
    //k, U ile baslayan kullanıcı sayısından fazlaysa diye kontrol ediyorum.
    if (k > user_count)
    {
        printf("You entered '%d', but there are only '%d' users..\n", k, user_count);
        k = user_count;
    }
}

```

```

//eger, oneri yapilacak kullanıcı NU degil de U ile basliyorsa, indisi k'dan yani U ile
baslayan kullanıcı sayisindan kucuktur. o durumda tasma olmamasi yani NU'ların isleme dahil
edilmemesi yani
//uc deger istisnasi olusmaması için k'yi bir azaltiyorum.
if (k == user_count && user_index < k)
{
    k--;
}
return kMostSimilars(users, user_count, n_user_count, book_count, user_name, user_index, k);
}

//Kullaniciya kitap onerisi yapildiktan sonra bu oneri ve en benzer olduğu k kullanıcı dosyaya
yazilir.
void saveToFile(USER_DATA* users, SIMILARITY_DATA* similarities, char* user_name, char*
most_suggested_book, int k)
{
    FILE* fp;
    char* file_name;
    int i = 0;
    char number[16];
    //_itoa(k, number, 10);
    snprintf(number, sizeof(number), "%d", k);
    file_name = (char*)calloc(strlen(user_name) + 1 + strlen("_kitap_onerisi_ve_en_benzer_") +
strlen("_kullanici.csv") + strlen(number) + strlen(BOOK_SUGGESTIONS_DIRECTORY) + strlen("./"),
sizeof(char));
    file_name = strcpy(file_name, "./");
    file_name = strcat(file_name, BOOK_SUGGESTIONS_DIRECTORY);
    file_name = strcat(file_name, "/");
    file_name = strcat(file_name, user_name);
    file_name = strcat(file_name, "_kitap_onerisi_ve_en_benzer_");
    file_name = strcat(file_name, number);
    file_name = strcat(file_name, "_kullanici.csv");
    fp = fopen(file_name, "w");
    if (fp == NULL)
    {
        printf("Can't open output file for writing..\nMake sure the following file is not open
in your computer right now..Returning..\n");
        printf("%s\n", file_name);
        return;
    }
    fprintf(fp, "User name%c%s\n", CSV_SEPERATOR, user_name);

    fprintf(fp, "%d most similar users%c", k, CSV_SEPERATOR);
    for (i = k - 1; i > -1; i--)
    {
        fprintf(fp, "%s%c", users[similarities[i].user_index].user_name, CSV_SEPERATOR);
    }
    fprintf(fp, "\n");
    fprintf(fp, "Suggested book%c%s\n", CSV_SEPERATOR, most_suggested_book);
    fclose(fp);
    free(file_name);
}

//kitap onerisi yapan fonksiyon
void suggestBook(USER_DATA* users, char** book_names, int user_count, int n_user_count, int
book_count)
{
    int i; //dongu degiskeni
    int j; //dongu degiskeni
    int k; //en benzer olarak bulunacak k kisi degiskeni
    int x; //dongu degiskeni
    int* unread_book_indexes; //oneri yapilacak kisinin okumadigi kitapların indisleri bu dizide
tutulur
    int unread_book_count = 0; //oneri yapilacak kisinin okumadigi kitap sayisi
    int user_index = 0; //oneri yapilacak kisinin indisi, scanf ile bu degiskende tutulur

```

```

    int other_user_read_book_count = 0; //k adet kisinin her biri icin hesaplanacak olan, o
    kisinin okudugu kitap sayisini tutan degisken
    double mean = 0; //oneri yapilacak kisinin verdigi puanlari ortalama
    double other_users_mean = 0; //diger kullanicinin puan ortalama
    BOOK_SUGGESTION_DATA* preds; //kitaplara verilecek tahmini puanlari tutuldu
    double current_pred = 0; //dongu icinde o anki kitaba verilecek tahmini puan bu degisken
    tutulur.
    double sum1 = 0; //paydaki toplama isleminin sonucu tutulur
    double sum2 = 0; //paydadaki toplama isleminin sonucu tutulur
    char user_name[MAX_WORD_SIZE]; //kullanici adi tutulur
    SIMILARITY_DATA* similarities; //benzerlikler dizisi.

    printf("Enter user name to suggest a book: ");
    scanf("%s", user_name);

    //girilen kullanici adina gore kullanici indisi bulunur. bulunamazsa fonksiyondan donulur.
    user_index = getUserIndex(users, user_count + n_user_count, user_name);
    if (user_index == -1)
    {
        return;
    }

    printf("Enter k: ");
    scanf("%d", &k);
    //k negatif girilmesi durumu kontrol ederiz.
    if (k < 1)
    {
        printf("k must be positive..Returning..\n");
        return;
    }

    //k, U ile baslayan kullanici sayisindan fazlaysa diye kontrol ediyorum.
    if (k > user_count)
    {
        printf("You entered '%d', but there are only '%d' users..\n", k, user_count);
        k = user_count;
    }

    //for icinde, oneri yapilacak kisinin okumadigi kitap sayisi ve indisleri ilgili yerlere
    atanir.
    unread_book_indexes = (int*)calloc(1, sizeof(int));
    for (i = 0; i < book_count; i++)
    {
        if (users[user_index].book_ratings[i] == 0)
        {
            unread_book_count++;
            unread_book_indexes = (int*)realloc(unread_book_indexes, sizeof(int) *
unread_book_count);
            unread_book_indexes[unread_book_count - 1] = i;
        }
    }
    //unread_book_indexes = (int*)realloc(unread_book_indexes, sizeof(int) * unread_book_count);

    //okumadigi kitap yoksa fonksiyondan donulur.
    if (unread_book_count == 0)
    {
        printf("' %s' has read all books...Returning..\n", user_name);
        return;
    }

    //tahmini puan dizisine okumadigi kitap sayisi kadar elemanlik yer acilir.
    preds = (BOOK_SUGGESTION_DATA*)calloc(unread_book_count, sizeof(BOOK_SUGGESTION_DATA));

    //eger, oneri yapilacak kullanici NU degil de U ile basliyorsa, indisi k'dan yani U ile
    baslayan kullanici sayisindan kucuktur. o durumda tasma olmamasi yani NU'larin isleme dahil
    edilmemesi yani

```

```

//uc deger istisnasi olusmamasini icin k'yi bir azaltiyorum.
if (k == user_count && user_index < k)
{
    k--;
}
//benzerlikler dizisi kmostsimilars fonksiyonu ile bulunur
similarities = kMostSimilars(users, user_count, n_user_count, book_count, user_name,
user_index, k);

//kullanıcının verdiği puanların ortalaması hesaplanır.
for (x = 0; x < book_count; x++)
{
    mean += users[user_index].book_ratings[x];
}
mean /= ((double)book_count - unread_book_count);

//oneri yapılacak kullanıcının okumadığı her bir kitap için bu dongu çalışır.
for (i = 0; i < unread_book_count; i++)
{
    sum1 = 0; //formuldeki pay kısmı hesaplanırken kullanılır.
    sum2 = 0; //formuldeki payda kısmı hesaplanırken kullanılır.
    //k sayısı kadar yani benzerlik hesaplanan kişi sayısı kadar bu dongu çalışır.
    for (j = 0; j < k; j++)
    {
        other_users_mean = 0;
        other_user_read_book_count = 0;
        //diğer kullanıcının puan ortalaması hesaplanır ve okuduğu kitap sayısı
hesaplanır.
        for (x = 0; x < book_count; x++)
        {
            other_users_mean += users[similarities[j].user_index].book_ratings[x];
            if (users[similarities[j].user_index].book_ratings[x] != 0)
            {
                other_user_read_book_count++;
            }
        }
        //diğer kullanıcının puan ortalaması hesaplanır.
        other_users_mean /= other_user_read_book_count;

        //formuldeki pay kısmı hesaplanır
        sum1 += (similarities[j].similarity *
(users[similarities[j].user_index].book_ratings[unread_book_indexes[i]] - other_users_mean));
        //formuldeki payda kısmı hesaplanır
        sum2 += (similarities[j].similarity);
    }
    //mevcut kitaba verilecek tahmini puan hesaplanır
    current_pred = mean + sum1 / sum2;
    //ilgili atamalar yapılır.
    preds[i].book_index = unread_book_indexes[i];
    preds[i].predicted_rating = current_pred;
}

//oneriler dizisi sıralanır. ciktida duzugun gozukmesi icin.
sortBookSuggestionArray(preds, unread_book_count);

//yazdırma işlemleri.
printf("Book suggestions and predicted ratings for '%s'\n", user_name);
for (i = 0; i < unread_book_count; i++)
{
    printf("%-2d| %-20s| %10f\n", i + 1, book_names[preds[i].book_index],
preds[i].predicted_rating);
}
printf("The book to be suggested is: ");

//onerilen kitabın rengi farklı yapılır. güzel gozukmesi icin.
#ifdef _WIN32

```

```

        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
        printf("%s", book_names[preds[0].book_index]);
    #elif _WIN64
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
        printf("%s", book_names[preds[0].book_index]);
    #else
        printf("\033[0;32m%s", book_names[preds[0].book_index]);
        printf("\033[0;37m");
    #endif
    #ifdef _WIN32
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN | FOREGROUND_RED |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
    #elif _WIN64
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN | FOREGROUND_RED |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
    #else
    #endif

    printf("\n");

    free(unread_book_indexes);
    saveToFile(users, similarities, user_name, book_names[preds[0].book_index], k);
}

int main()
{
    FILE* fp; //okunacak dosya pointeri
    USER_DATA* users; //kullanıcılar dizisi
    char** book_names; //kitap isimleri matrisi
    int book_count = 0; //kitap sayısı hesaplanır
    int user_count = 0; //kullanıcı sayısı hesaplanır
    int n_user_count = 0; //NU ile başlayan kullanıcı sayısı hesaplanır.
    int opCode = 0; //yapılacak işlemin kodu

    setlocale(LC_ALL, "Turkish");
    //konsol rengi ayarlanır
    #ifdef _WIN32
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN | FOREGROUND_RED |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
    #elif _WIN64
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN | FOREGROUND_RED |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
    #else
        printf("\033[0;37m");
    #endif

    //dosya açma işlemi yapılır
    fp = fopen(RECOMMENDATION_FILE_NAME, "r");
    if (fp == NULL)
    {
        //dosya bulunamazsa hata verilip program kapatılır.
        printf("Can't open the file '%s'.. Exiting..", RECOMMENDATION_FILE_NAME);
        exit(-1);
    }

    #ifdef _WIN32
        _mkdir(BOOK_SUGGESTIONS_DIRECTORY, 0777);
    #elif _WIN64
        _mkdir(BOOK_SUGGESTIONS_DIRECTORY, 0777);
    #elif __linux__
        mkdir(BOOK_SUGGESTIONS_DIRECTORY, 0777);
    #endif

    //kitap isimleri okunur

```



```

book_names = populateBookNamesArray(fp, &book_count);
//printBookNames(book_names, book_count);

//kullaniciilar dizisi olusturulur.
users = populateUserArray(fp, book_count, &user_count, &n_user_count);
//printUserArray(users, user_count, book_count);

do
{
    printf("0-Exit\n1-Calculate two users' similarity\n2-Calculate k most similar users to
a user\n3-Suggest new book to user\n\nEnter the operation code: ");
    //printf("0-Exit\n1-Iki kullanicinin benzerligini hesapla\n2-Bir kullaniciya en benzer
k adet kullanici hesapla\n3-Kitap oner\n\nIslem kodunu giriniz: ");
    scanf("%d", &opCode);

    switch (opCode)
    {
        case 0:
            break;
        case 1:
            //1 girildiyse iki kullanici benzerligi hesaplanir
            twoUserSimilarity(users, user_count, n_user_count, book_count);
            break;
        case 2:
            //2 girildiyse en benzer k kullanici hesaplanir
            kMostSimilarsEntry(users, user_count, n_user_count, book_count);
            break;
        case 3:
            //3 girildiyse kitap onerisi yapilir.
            suggestBook(users, book_names, user_count, n_user_count, book_count);
            break;
        default:
            printf("Hatali giris yaptiniz..!!\n");
            break;
    }
    printf("\n\n");
} while (opCode);

fclose(fp);
return 0;
}

```