2020-2021 Güz Yarıyılı
Algoritma Analizi
Dönem Projesi

Ders Yürütücüleri
Doç. Dr. M. Elif KARSLIGİL
Dr. Öğr. Üyesi M. Amaç GÜVENSAN

BURAK BOZ
18011706

Problem: Kitap Öneri Sistemi. 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 çalışmada işbirlikçi filtre (collaborative filtering) yöntemi ile bir kişinin önceki seçimlerine bakarak yeni kitap öneren bir sistem tasarlanacak ve gerçeklenecektir.

Çözüm: Programa başlarken öncelikle "RecomendationDataSet.csv" dosyasından kullanıcıların verileri alınmıştır. Bu csv dosyasının ilk satırı kitap isimlerinden, geri kalan kısımlar ise kullanıcılardan oluşturulmuştur.

Veriler program içerisinde linkli liste yapısında saklanmıştır.

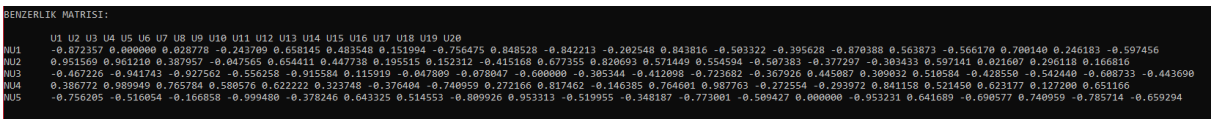
- Kitaplar için KITAP isminde ve sadece kitap ismini tutan bir struct linkli listesi (**8 Kitap olduğu varsayılmıştır.**)
- Kullanıcılar için KISI isminde, kişinin adını ve kitaplara verdiği puanları dizi şeklinde tutan bir linkli liste tasarlanmıştır. Kullanıcı adı U ile başlayan kullanıcılar farklı bir listede, NU ile başlayan kullanıcılar ise farklı bir linkli listede saklanmıştır.

Program başlatıldığı anda gerekli değişken tanımlamaları yapılmış ve ardından linkli listeler, csv dosyasındaki veriler ile doldurulmuştur. Veriler okunduktan sonra bir benzerlik matrisi oluşturulmuştur. Bu matris NU kişileri kadar satırdan ve U kişileri kadar sütundan oluşmaktadır. Bu matris kişiler arasındaki benzerlikleri tutmaktadır. Örneğin NU1 kişisinin U15 isimli kişi ile arasındaki benzerliğe ulaşmak için simMatris[0][14] adresindeki veriye ulaşmak yeterlidir.

Matris doldurulurken;

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

Formülünden yararlanılmıştır. Burada a NU kişisini, b ise U kişisini temsil etmektedir. Matrisin dolu hali şu şekildedir;



Matris doldurulduktan sonra kullanıcıdan hangi kişiye ait benzerlik bilgisine ulaşmak istediği bilgisi alınır. Sonra en yakın k kişi kadar kişinin listelenebilmesi için k sayısı alınır. Bu veriler alındıktan sonra matris zaten dolu olduğu için matrisin NU satırının en büyük k adet elemanı sırası ile yazdırılır. Buradaki sıralama, dizideki en büyük sayıyı bulma mantığı ile yazdırılmaktadır. Dolayısıyla sıralama algoritmalarından daha az karmaşıklık ile bu problem çözülmüştür.

NU kişisine benzer olduğu kişiler arasından kitap önerebilmek için;

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Formülünden faydalanılmıştır. Burada a NU kişisini, p okunmayan kitabı ve b ise benzer olan k kişinin indeksini temsil etmektedir. Önce kişinin okumadığı kitaplar belirlenir. Bu kitaplar bir dizide tutulduktan sonra, benzer kişilerin bu kitaba kaç puan verdiği bilgisinden faydalanılarak kullanıcı için tahmini bir puan oluşturulur. Oluşturulan puanlar neticesinde okunmayan kitaplar için en yüksek tahmini puanı alan kitap kullanıcıya önerilir.

EKRAN ÇIKTILARI

Not: Ekran çıktılarında ödev metninde belirtildiği üzere k sayısı her çıktı için 3 girilmiştir fakat farklı değer girilebilir.

NU1 Kullanıcısı İçin:

```
Programi kapatmak icin 'cikis' giriniz.
Kitap onerisi yapilacak kullanici adini girin (Ornek: NU1):NU1
En yakin k kisinin bulunmasi icin k degerini giriniz:3


En Yakin Kullanicilar:
U9 : 0.848528
U12 : 0.843816
U18 : 0.700140
OKUNMAYAN KITAPLARA VERILEBILECEK TAHMINI PUANLAR:
THE DA VINCI CODE isimli kitap icin tahmini puan : 4.833333
RUNNY BABBIT isimli kitap icin tahmini puan : 3.284463

Kisiye onerilen kitap = THE DA VINCI CODE
Bellek temizlendi

--------------------------------
Process exited after 2.945 seconds with return value 0
Press any key to continue . . .
```

NU2 Kullanıcısı İçin:

```
Programi kapatmak icin 'cikis' giriniz.
Kitap onerisi yapilacak kullanici adini girin (Ornek: NU1):NU2
En yakin k kisinin bulunmasi icin k degerini giriniz:3


En Yakin Kullanicilar:
U2 : 0.961210
U1 : 0.951569
U11 : 0.820693
OKUNMAYAN KITAPLARA VERILEBILECEK TAHMINI PUANLAR:
TRUE BELIEVER isimli kitap icin tahmini puan : 3.200000
THE KITE RUNNER isimli kitap icin tahmini puan : 2.820344
HARRY POTTER isimli kitap icin tahmini puan : 2.768017

Kisiye onerilen kitap = TRUE BELIEVER
Bellek temizlendi

--------------------------------
Process exited after 3.649 seconds with return value 0
Press any key to continue . . .
```

NU3 Kullanıcısı İçin:

```
Programi kapatmak icin 'cikis' giriniz.
Kitap onerisi yapilacak kullanici adini girin (Ornek: NU1):NU3
En yakin k kisinin bulunmasi icin k degerini giriniz:3


En Yakin Kullanicilar:
U16 : 0.510584
U14 : 0.445087
U15 : 0.309032
OKUNMAYAN KITAPLARA VERILEBILECEK TAHMINI PUANLAR:
THE WORLD IS FLAT isimli kitap icin tahmini puan : 1.438836
MY LIFE SO FAR isimli kitap icin tahmini puan : 1.404569

Kisiye onerilen kitap = THE WORLD IS FLAT
Bellek temizlendi

--------------------------------
Process exited after 3.193 seconds with return value 0
Press any key to continue . . .
```

NU4 Kullanıcısı İçin:

```
Programi kapatmak icin 'cikis' giriniz.
Kitap onerisi yapilacak kullanici adini girin (Ornek: NU1):NU4
En yakin k kisinin bulunmasi icin k degerini giriniz:3


En Yakin Kullanicilar:
U2 : 0.989949
U13 : 0.987763
U16 : 0.841158
OKUNMAYAN KITAPLARA VERILEBILECEK TAHMINI PUANLAR:
THE TAKING isimli kitap icin tahmini puan : 2.150650
RUNNY BABBIT isimli kitap icin tahmini puan : 2.247067

Kisiye onerilen kitap = RUNNY BABBIT
Bellek temizlendi

--------------------------------
Process exited after 4.063 seconds with return value 0
Press any key to continue . . .
```

NU5 Kullanıcısı için:

```
Programi kapatmak icin 'cikis' giriniz.
Kitap onerisi yapilacak kullanici adini girin (Ornek: NU1):NU5
En yakin k kisinin bulunmasi icin k degerini giriniz:3


En Yakin Kullanicilar:
U9 : 0.953313
U18 : 0.740959
U6 : 0.643325
OKUNMAYAN KITAPLARA VERILEBILECEK TAHMINI PUANLAR:
TRUE BELIEVER isimli kitap icin tahmini puan : 4.099034
THE KITE RUNNER isimli kitap icin tahmini puan : 2.623434
HARRY POTTER isimli kitap icin tahmini puan : 2.841302

Kisiye onerilen kitap = TRUE BELIEVER
Bellek temizlendi

--------------------------------
Process exited after 10.39 seconds with return value 0
Press any key to continue . . .
```

KULLANILAN FONKSİYONLAR

void parse(char * ,int);

inputlar: Bir satır metin, okunan satırın indeksi

Açıklama: İnput dosyasındaki her bir satırı alıp “;” karakterine göre bölmektedir. Eğer satırın indeksi 0 ise bu satırda kitap isimleri bulunmaktadır. Buna göre kitap eklemesi yapmaktadır. Eğer satır numarası 0’dan büyük ise kullanıcı adına göre ekleme yapmaktadır. Kullanıcı adı U ile başlıyorsa normal kullanıcı, NU ile başlıyorsa test kullanıcısı eklemektedir.

void kisiSonaEkle(char * , int []);
struct KISI* kisiOlustur(char * , int []);

inputlar: Kişi adı,Kişinin kitaplara verdiği puanlar

Açıklama: Linkli listeye yeni bir kişi eklemek için kullanılan fonksiyonlardır.

void kisileriYaz();

Açıklama: Kişilerin bulunduğu linkli listedeki verileri yazdırmak için kullanılır.

void sim(int , int ,struct KISI *, struct KISI *);

İnputlar: matrisin satır indeksi, matrisin sütun indeksi, NU kişisinin adresi, U kişisinin adresi

Açıklama: Kişiler arasındaki benzerliği hesaplayıp benzerlik matrisinin doldurulmasını sağlayan fonksiyondur.

void simMatrisYazdir(int ,int);

İnputlar: matrisin satır sayısı, matrisin sütun sayısı

Açıklama: Benzerlik matrisini ekrana yazdırmak için kullanılır.

struct KISI *kisiGetir(char * kadi)

İnputlar: kullanıcı adı

Açıklama: Kullanıcı adı girilen kişinin adresini döndüren fonksiyondur.

int *kKisiBul(struct KISI *kisi,int k,int usrAdet)

İnputlar: NU kişisinin adresi, benzerliği bulunacak k adet kişi, sistemdeki U kişi adedi

Açıklama: Benzerlik matrisindeki en çok benzeyen k adet kişiyi bulan ve bir diziye bu kişilerin matristeki sıralarını dizi şeklinde döndüren fonksiyondur. Örneğin NU1 kişisine en çok benzeyen 3 kişi girildiğinde bu fonksiyon [15,4,8] verisini döndürmektedir. Bu veri “Benzerlik matrisindeki NU kullanıcının bulunduğu satırdaki U16,U5,U9 kişileri, sırasıyla bu NU kullanıcısına en çok benzeyen kişilerdir.” anlamına gelmektedir.

int predict(struct KISI *,int ,int *);
İnputlar: NU kişisinin adresi, Benzerliği hesaplanacak k kişi sayısı, en benzer k kişinin benzerlik matrisindeki sıralarının bulunduğu dizi
Açıklama: Bu fonksiyon kitap önerme işlemini gerçekleştiren fonksiyondur. Verilen formüldeki işlemleri yapar ve okunmayan kitaplar için birer float sayı değeri üretir. Bu üretilen sayı değerlerinden en yüksek olan kitabın sırasını int olarak döndürmektedir. Bu sıra numarası KITAP linkli listesindeki sıraya tekabül etmektedir. Örneğin buradan döndürülen 3 sayısı KITAP linkli listesinde "THE WORLD IS FLAT" isimli kitaba denk geliyorsa, kullanıcıya önerilecek olan en yüksek tahmini puanlı kitap budur.

void kitapSonaEkle(char *);
struct KITAP* kitapOlustur(char *);
İnputlar: Kitap adı
Açıklama: Kitap linkli listesine kitap eklemeye yarayan fonksiyondur.

void kitaplariYaz();
Açıklama: Kitap linkli listesinde bulunan kitapları ekrana yazdıran fonksiyondur.

struct KITAP* kitapGetir(int);
İnputlar: Getirilecek kitabın sırası
Açıklama: Bu fonksiyon sıra numarası girilen kitabın adresini döndürmektedir. Bu sayede kitabın isim bilgisine ulaşılabilmektedir.

Kodlar:

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

struct KISI//Kişiler
{
	char ad[10];
	int kitapPuanlari[8];
	struct KISI *next;
};

struct KITAP//Kitaplar
{
	char ad[100];
	struct KITAP *next;
};

struct KISI *kisiHead;		//U ile başlayan kişilerin linkli listesinin başlangıç adresini tutacaktır.
struct KISI *tKisiHead;		//NU ile başlayan kişilerin linkli listesinin başlangıç adresini tutacaktır.
double **simMatrix;			//Benzerlik matrisinin başlangıç adresini tutacaktır.
struct KITAP *kitapHead;//Kitapların saklandığı linkli listenin başlangıç adresini tutacaktır.

void parse(char * ,int);		//csv dosyasından veri okuyan fonksiyondur.
void kisiSonaEkle(char * , int []);		//Kişi linkli listesinin sonuna yeni kişi ekler.
struct KISI* kisiOlustur(char * , int []);	//Yeni bir kişi oluşturmak için bellekte yer açar.
void kisileriYaz();	//Kişilerin bulunduğu listelerin yazdırılması için kullanılır.
int userSayac();	//U ile başlayan kullanıcıların adedini döndürür.
int nUserSayac();	//NU ile başlayan kullanıcıların adedini döndürür.
void sim(int , int ,struct KISI *, struct KISI *);	//Kişiler arasındaki benzerlikleri hesaplar ve benzerlik matrisi oluşturur.
void simMatrisYazdir(int ,int );		//Benzerlik matrisinin ekrana yazdırılmasını sağlar.
struct KISI *kisiGetir(char * );//Kullanıcı adı girilen kişinin adresini döndürür.
int *kKisiBul(struct KISI *,int ,int );		//Benzerlik matrisindeki en benzer k adet kişiyi bulur.
int predict(struct KISI *,int ,int * );		//NU kişisinin okumadığı kitaplar için tahmini puan oluşturur ve en yüksek puanlı kitabın sırasını döndürür.
void kitaplariYaz();	//Kitaplar linkli listesindeki elemanların ekrana yazılmasını sağlar.
void kitapSonaEkle(char * );	//Kitaplar linkli listesinin sonuna yeni bir kitap ekler.
struct KITAP* kitapOlustur(char * );	//Listeye eklenecek yeni kitap için bellekte yer açar ve aderesi döndürür.
struct KITAP* kitapGetir(int);//Sırası girilen kitabın adresini döndürür.


int main()
{
	kisiHead=NULL;
	tKisiHead=NULL;
	kitapHead=NULL;
	int nUserAdet;//NU ile başlayan kullanıcı adedini saklayacaktır.
	int userAdet;//U ile başlayan kullanıcı adedini saklayacaktır.
	char kullaniciAdi[10];//hakkında arama yapılacak NU kullanıcısının adını saklayacaktır.
	struct KISI *arananKisi=NULL;//kullaniciAdi girilen kişinin adresini saklayacaktır.
	int k;//En benzer K kişi
	int satirNo=0;//csv den okuma yaparken okunan satırın indeksini tutacaktır.
	int i = 0,j=0;//Döngülerde kullanılacak
	int menuDurum=1;//Menüde doğru giriş yapılana kadar kontrol sağlamak için kullanılacaktır.
	char cikis[]="cikis";//Menüde programın kapatılmasını komutunun kontrol edilmesi için kullanılacaktır.
```

```
	int onerilecekKitapSirasi=0;//Önerilecek kitabın sırasını saklayacaktır.

	FILE *dosya = fopen("RecomendationDataSet.csv","r");
	if(dosya==NULL)
	{
		perror("Dosya acilamadi.\n");
	}
	char line[1000];
	while(fgets(line,sizeof(line),dosya))
	{
		char *token;
		token=strtok(line,",");
		while(token != NULL)
		{
			int len = strlen(token);
			token[len-1]=';';
			token[len]='\0';
			parse(token,satirNo);
			token = strtok(NULL,",");
		}
		satirNo++;
	}
	kitaplariYaz();
	kisileriYaz();
  nUserAdet = nUserSayac();
  userAdet = userSayac();
	simMatrix=(double **)malloc(nUserAdet*sizeof(double*));//Benzerlik matrisi csv den okunan eleman sayılarına
göre oluşturuluyor.
	if(simMatrix!=NULL)
	{
		for(i=0;i<nUserAdet;i++)
		{
			simMatrix[i]=(double *)malloc(userAdet*sizeof(double));
			if(simMatrix[i]==NULL){//Eğer bellekte yer yoksa hata döndürülecek.
				printf("Tip -1 hata olustu");
				return -1;
			}
		}
	}
  printf("Sim matrix olusturuluyor\n");
  struct KISI *tmpTestKisi = tKisiHead;
  for(i=0;i<nUserAdet;i++)
  {
	struct KISI *tmpKisi = kisiHead;
	for(j=0;j<userAdet;j++)
	{
		sim(i,j,tmpTestKisi,tmpKisi);
		tmpKisi = tmpKisi->next;
		}
		tmpTestKisi=tmpTestKisi->next;
	}
  simMatrisYazdir(userAdet,nUserAdet);
  printf("\n\n");
  do{//Menü
	printf("Programi kapatmak icin 'cikis' giriniz.\n");
	printf("Kitap onerisi yapilacak kullanici adini girin (Ornek: NU1):");
	gets(kullaniciAdi);
	if(strcmp(kullaniciAdi,cikis)==0)//Programın işlem yapmadan kapatılmasını sağlayaktır.
	{
```

```
                menuDurum=0;
                }
                else
                {
                        arananKisi=kisiGetir(kullaniciAdi);//Girilen kullanıcı adına ait bir kullanıcı gerçekten varsa adresi
dönecektir.
                        if(arananKisi==NULL)
                {
                        printf("Gecersiz kullanici adi. Lutfen tekrar deneyiniz. Programi kapatmak icin 0 giriniz.\n");
                        }
                }

        }while(arananKisi==NULL && menuDurum==1);//Aksi durumlarda tekrardan kullanıcı adı istenecektir.

        if(menuDurum!=0)//Eğer menüden doğru şekilde çıkıldıysa işlemler gerçekleşecektir.
        {
                printf("En yakin k kisinin bulunmasi icin k degerini giriniz:");
                scanf("%d",&k);
                int *kSiralari;


                kSiralari=kKisiBul(arananKisi,k,userAdet);//En yakın k kişinin indeksleri kSiralari dizisinde saklanacaktır.
                printf("OKUNMAYAN KITAPLARA VERILEBILECEK TAHMINI PUANLAR:\n");
                onerilecekKitapSirasi = predict(arananKisi,k,kSiralari);//En yüksek tahmini puanı alan kitap
önerilecekKitapSirasi isimli değişkende saklanacaktır.

                struct KITAP *tmpKitapOner = kitapGetir(onerilecekKitapSirasi);//Önerilecek kitabın adresi  linkli listeden
gelir.
                printf("\nKisiye onerilen kitap = %s\n",tmpKitapOner->ad);
        }
        else
        {
                printf("Program kapatiliyor...\n");
        }
```

```
  //TEMİZLİK | Bellekteki ayrılan yerlerin tekrar işletim sistemine bırakılması.
        struct KISI * tmp = kisiHead;
  while(kisiHead != NULL)
  {
        tmp = kisiHead->next;
        free(kisiHead);
        kisiHead = tmp;
        }

        tmp = tKisiHead;
  while(tKisiHead != NULL)
  {
        tmp = tKisiHead->next;
        free(kisiHead);
        tKisiHead = tmp;
        }

        struct KITAP *tmp2 = kitapHead;
  while(tKisiHead != NULL)
```

```
  {
        tmp2 = kitapHead->next;
        free(kitapHead);
        kitapHead = tmp2;
        }

        for(i=0;i<nUserAdet;i++)
        {
                free(simMatrix[i]);
        }
        free(simMatrix);
        printf("Bellek temizlendi\n");

  return 0;
}

void parse(char * metin,int satirNo)
{
        /*
        İnputlar:  csv den gelen bir satır, bu satırın numarası
        Açıklama:       csv den okunan satırın gerekli linkli listeye eklenmesini sağlar.
        */
        int cursor=0;
        char yeniMetin[1000];
        int i=0,j=0;
        while(metin[i] != '\0')
        {
                if(metin[i]==';')
                {
                        if(metin[i+1]==';')
                        {
                                yeniMetin[cursor]=metin[i];
                                cursor++;
                                yeniMetin[cursor]=' ';
                                cursor++;
                        }
                        else
                        {
                                yeniMetin[cursor]=metin[i];
                                cursor++;
                        }
                }
                else
                {
                        yeniMetin[cursor]=metin[i];
                        cursor++;
                }

                i++;
        }
        yeniMetin[cursor]='\0';
        char delim[] = ";";
        int sayac = 0;
        char *ptr = strtok(yeniMetin, delim);

        if(satirNo!=0)
        {
                struct KISI yeniKisi;
                while(ptr != NULL)
```

```
		{

			if(sayac==0)
			{
				strcpy(yeniKisi.ad,ptr);
			}
			else
			{
				if(strcmp(ptr," ")==0 || strlen(ptr)==0)
				{
					yeniKisi.kitapPuanlari[sayac-1]=0;

				}
				else
				{

					yeniKisi.kitapPuanlari[sayac-1]=atoi(ptr);
				}
			}
			ptr = strtok(NULL, delim);
			sayac++;
		}

		kisiSonaEkle(yeniKisi.ad,yeniKisi.kitapPuanlari);

	}
	else
	{
		kitapSonaEkle(yeniMetin);
		//printf("Bu bir kitaptir : %s\n",yeniMetin);
	}

}

struct KISI* kisiOlustur(char * kisiAd, int puanlar[8])
{
	/*
	inputlar:  kişinin adı, kişinin kitaplara verdiği puanlar
	Açıklama:		Yeni bir kişi eklemek için bellekte bu kişiye yer açar ve bu yerin adresini döndürür.
	*/
	int i = 0;
	struct KISI* yeniKisi = (struct KISI*) malloc(sizeof(struct KISI));
	strcpy(yeniKisi->ad,kisiAd);
	for(i = 0; i<8 ; i++)
	{
		yeniKisi->kitapPuanlari[i]=puanlar[i];
	}
	yeniKisi->next=NULL;

	return yeniKisi;
}


void kisiSonaEkle(char * kisiAd, int puanlar[]){
	/*
	İnputlar:  kişi adı,kişinin kitaplara verdiği puanlar
	Açıklama:		Yeni gelen kişiyi ilgili linkli listenin sonuna ekler
	*/
  struct KISI* sonaEklenecek = kisiOlustur(kisiAd,puanlar);
```

```
  if(kisiAd[0]=='N')//Eğer kişi adı N ile başlıyorsa NU kişilerinin tutulduğu linkli listeye ekler
  {
        if (tKisiHead == NULL)
          {
            tKisiHead = sonaEklenecek;
          }
          else
          {
                struct KISI* tmp = tKisiHead;
                while(tmp->next != NULL)
                {
                        tmp=tmp->next;
                        }
                        tmp->next=sonaEklenecek;
          }
        }
        else if(kisiAd[0]=='U')//Eğer kişi adı U ile başlıyorsa U kişilerinin tutulduğu linkli listeye ekler
        {
                if (kisiHead == NULL)
          {
            kisiHead = sonaEklenecek;
          }
          else
          {
                struct KISI* tmp = kisiHead;
                while(tmp->next != NULL)
                {
                        tmp=tmp->next;
                        }
                        tmp->next=sonaEklenecek;
          }
        }


}

void kisileriYaz()     //Kişilerin bulunduğu linkli listedeki verileri(Kişi adı ve kitapalra verdiği puanlar) ekrana yazar.
{
        int j = 0;
        struct KISI *tmp = kisiHead;
        printf("NORMAL KULLANICILAR:\n");
        while(tmp != NULL)
        {
                printf("Kisi ad:%s\n",tmp->ad);
                for(j=0;j<8;j++)
                {
                        printf("%d . kitap puani:%d\n",j+1,tmp->kitapPuanlari[j+1]);
                }
                printf("\n====================\n");
                tmp = tmp->next;
        }

        tmp = tKisiHead;
        printf("TEST KULLANICILARI:\n");
                while(tmp != NULL)
        {
                printf("Kisi ad:%s\n",tmp->ad);
                for(j=0;j<8;j++)
                {
```

```
                        printf("%d . kitap puani:%d\n",j+1,tmp->kitapPuanlari[j+1]);
                }
                printf("\n====================\n");
                tmp = tmp->next;
        }
}

int userSayac()//Kaç tane U isimli kullanıcı olduğunu döndürür.
{
        int sayac=0;
        struct KISI *tmp = kisiHead;
        while(tmp!=NULL)
        {
                sayac++;
                tmp= tmp->next;
        }
        return sayac;
}

int nUserSayac()//Kaç tane NU isimli kullanıcı olduğunu döndürür.
{
        int sayac=0;
        struct KISI *tmp = tKisiHead;
        while(tmp!=NULL)
        {
                sayac++;
                tmp= tmp->next;
        }
        return sayac;
}

void sim(int i, int j,struct KISI *test, struct KISI *kisi)
{
        /*
        inputlar:  matris satırı,matris sütunu, NU kişinin adresi, U kişinin adresi
        Açıklama:       Bu iki kişi arasındaki benzerliği hesaplar ve benzerlik matrisinin [i][j] gözüne bu hesaplanan
değeri yazar.
        X = [Rap-Ra]
        Y = [Rbp-Rb]
        a=Kök(Rap-Ra)^2)
        b=Kök(Rbp-Rb)^2)
        */
        double x=0,y=0;
        double a=0,b=0;
        int k=0,l=0;//Döngü için kullanılacak
        double rA=0,rB=0;
        int ortakSayac=0;//Ortak okunan kitap sayısı
        int aSayac=0;
        int bSayac=0;
        double sonuc=0;//Hesaplanan sonuç

        for(k = 0;k<8;k++)
        {
                if(test->kitapPuanlari[k]!=0)
                {
                        rA+=test->kitapPuanlari[k];
                        aSayac++;
                        if(kisi->kitapPuanlari[k]!=0)
                        {
```

```
                                        ortakSayac++;
                                }
                        }
                        if(kisi->kitapPuanlari[k]!=0)
                        {
                                bSayac++;
                                rB+=kisi->kitapPuanlari[k];
                        }
                }
                rA=rA/aSayac;
                rB=rB/bSayac;

                for(k = 0;k<8;k++)
                {
                        if(test->kitapPuanlari[k]!=0)
                        {
                                if(kisi->kitapPuanlari[k]!=0)
                                {
                                        x+=((test->kitapPuanlari[k]-rA)*(kisi->kitapPuanlari[k]-rB));
                                        a+=pow((test->kitapPuanlari[k]-rA),2);
                                        b+=pow((kisi->kitapPuanlari[k]-rB),2);
                                }
                        }
                }
                a=sqrt(a);
                b=sqrt(b);

                sonuc = (x)/(a*b);
                simMatrix[i][j]=sonuc;
                printf("%s test kisisi ile %s kisi benzerligi :%lf\n",test->ad,kisi->ad,sonuc);
}

void simMatrisYazdir(int uSay,int tSay)
{
                /*
                inputlar:  U sayısı,NU sayısı
                Açıklama:               Oluşturulan benzerlik matrisini ekrana yazdırır.
                */
                printf("\nBENZERLIK MATRISI:\n\n");
                int i=0,j=0;
                struct KISI *tmpT = tKisiHead;
                struct KISI *tmpKisi = kisiHead;
                printf("\t");
                for(i = 0; i<uSay;i++)
                {
                        printf("%s ",tmpKisi->ad);
                        tmpKisi=tmpKisi->next;
                }
                printf("\n");

                for(i = 0 ;i<tSay;i++)
                {
                        printf("%s\t",tmpT->ad);
                        struct KISI *tmpK = kisiHead;
                        for(j=0;j<uSay;j++)
                        {
                                printf("%lf ",simMatrix[i][j]);
                                tmpK = tmpK->next;
                        }
```

```
                printf("\n");
                tmpT = tmpT->next;
        }
}

struct KISI *kisiGetir(char * kadi)
{
        /*
        inputlar:  kullanıcı adı
        Açıklama:       Gelen kullanıcı adının adresini döndürür. Yoksa NULL döndürür.
        */
        struct KISI *tmp = tKisiHead;
        int kontrol=0;
        while(tmp != NULL && kontrol == 0)
        {
                if(strcmp(tmp->ad,kadi)==0)
                {
                        kontrol=1;
                }
                else
                {
                        tmp=tmp->next;
                }
        }
        if(kontrol==1)
        {
                return tmp;
        }
        else
        {
                return NULL;
        }
}

int *kKisiBul(struct KISI *kisi,int k,int usrAdet)
{
        /*
        inputlar:  NU kişisinin adresi, En yakın K kişi sayısı, U kullanıcı adedi
        Açıklama:       Gelen NU kişisine en yakın olan k adet kişiyi hesaplar ve bu kişilerin benzerlik matrisindeki
indekslerinin saklandığı bir dizinin başlangıç adresini döndürür
        */
        int i=0, j=0;
        float enb=-2.0;
        int sira=0;
        float tmpDizi[usrAdet];
        int kisiSirasi=0;
        int * kSiralar;
        struct KISI *tmp = tKisiHead;
        int donguKir=1;
        kSiralar = (int *) calloc(k,sizeof(int));
        while(tmp != NULL && donguKir == 1)
        {
                if(strcmp(kisi->ad,tmp->ad)==0)
                {
                        donguKir=0;
                }
                else
                {
                        kisiSirasi++;
```

```
                        tmp=tmp->next;
                }
        }

        for(i = 0;i<usrAdet;i++)
        {
                tmpDizi[i]=simMatrix[kisiSirasi][i];
        }

        for(j=0;j<k;j++)
        {
                sira=-1;
                enb=-3.0;
                for(i=0;i<usrAdet;i++)
                {
                        if(tmpDizi[i]>=enb)
                        {
                                enb=tmpDizi[i];
                                sira=i;
                        }
                }
                kSiralar[j]=sira;
                tmpDizi[sira]=-3.0;
        }
        printf("\n\n");
        printf("En Yakin Kullanicilar:\n");
        for(i=0;i<k;i++)
        {
                struct KISI *tmpYazdir = kisiHead;
                for(j=0;j<kSiralar[i];j++)
                {
                        tmpYazdir=tmpYazdir->next;
                }
                printf("%s - %lf\n",tmpYazdir->ad,simMatrix[kisiSirasi][kSiralar[i]]);
        }

        return kSiralar;
}


int predict(struct KISI *kisi,int k,int * kSiralar)
{
        /*
        inputlar:  NU kişisinin adresi, En yakın K kişi sayısı, En yakın k kişinin benzerlik matrisindeki indeksleri
        Açıklama:       Nu kişisinin okumadığı kitapları tespit eder. Bu kitaplara en yakın K kişinin verdiği puanları
tespit eder ve bu kişileri verdikleri puanlara
                        göre NU kişisinin okumadığı kitaplara verebileceği tahmini puanları hesaplar ve
ekrana yazdırır. Bu okunmayan kitaplar arasından en yüksek
                        puanı alan kitabın linkli listedeki sırasını döndürür.
        */
        float rA=0;
        int i=0,j=0;
        int okunanSayisi=0;//Ortalama hesaplmak için kullanılacak
        int okunMayanSayisi=0;//Okunmayan kitap sayısı
        float sonuc=0.0;//Her kitap için hesaplanan sonuç puan
        float pay=0.0,payda=0.0;//Sonucun hesaplanması için kullanılacak yardımcı değişkenler
        int nuSira =0;//NU kişisinin benzerlik matrisindeki satır indeksi
        int kontrol =0;//NU satır numarasını hesaplamak için yardımcı değişken
        float enbSonuc=-300.0;//Önerilecek kitaplar arasında alınan en yüksek puan bu değişkende saklanacaktır.
```

```
int onerilecekSira = 0;//En son en yüksek puanı alan kitabın sırası bu değişkende saklanacaktır.

struct KISI *tmpSira = tKisiHead;
while(tmpSira!=NULL && kontrol ==0)
{
	if(strcmp(tmpSira->ad,kisi->ad)==0)
	{
		kontrol =1;
	}
	else
	{
		tmpSira=tmpSira->next;
		nuSira++;
	}
}
for(i=0;i<8;i++)
{
	if(kisi->kitapPuanlari[i]!=0)
	{
		rA+=kisi->kitapPuanlari[i];
		okunanSayisi++;
	}
}
okunMayanSayisi=8-okunanSayisi;
rA=rA/((float)okunanSayisi);


int okunmayanSiralar[okunMayanSayisi]; // NU nun okumadığı kitapların indeksleri burada saklanacak.
int sayac=0;
for(i=0;i<8;i++)
{
	if(kisi->kitapPuanlari[i]==0)
	{
		okunmayanSiralar[sayac]=i;
		sayac++;
	}
}

int l=0,m=0;//Döngü için geçici değişkenler

for(i=0;i<okunMayanSayisi;i++)
{
	for(j=0;j<k;j++)
	{
		struct KISI *tmpKisiPuan = kisiHead;
		int kisiSirasi =0;
		for(l =0 ;l<kSiralar[j];l++)
		{
			tmpKisiPuan=tmpKisiPuan->next;
			kisiSirasi++;
		}
		if(tmpKisiPuan->kitapPuanlari[okunmayanSiralar[i]]!=0)
		{
			float rB=0.0;
			int bOkunanSayisi=0;//B nin okuduğu kitap sayısı
			int bOkunanToplam=0;//Rb
			for(m=0;m<8;m++)
			{
				if(tmpKisiPuan->kitapPuanlari[m]!=0)
```

```
                                        {
                                                bOkunanSayisi++;
                                                bOkunanToplam+=tmpKisiPuan->kitapPuanlari[m];
                                        }
                                }
                                rB=(float) bOkunanToplam/bOkunanSayisi;
                                pay += simMatrix[nuSira][kisiSirasi]*((tmpKisiPuan-
>kitapPuanlari[okunmayanSiralar[i]])-rB);
                                payda += simMatrix[nuSira][kisiSirasi];
                                sonuc = rA + (pay/payda);
                        }
                }
                printf("%s isimli kitap icin tahmini puan : %lf\n",kitapGetir(okunmayanSiralar[i]),sonuc);
                if(sonuc>=enbSonuc)
                {
                        enbSonuc=sonuc;
                        onerilecekSira=okunmayanSiralar[i];
                }
        }
        return onerilecekSira;

}


struct KITAP* kitapOlustur(char * kitapAd)
{
        /*
        inputlar:  eklenecek kitabın adı
        Açıklama:       Yeni eklenecek kitap için bellekte yer oluşturur ve adresini döndürür.
        */
        int i = 0;
        struct KITAP* yeniKitap = (struct KITAP*) malloc(sizeof(struct KITAP));
        strcpy(yeniKitap->ad,kitapAd);
        yeniKitap->next=NULL;

        return yeniKitap;
}


void kitapSonaEkle(char * kitapAd){
        /*
        inputlar:  eklenecek kitabın adı
        Açıklama:       Yeni gelen kitabı kitaplar linkli listesinin sonuna ekler
        */
  if(strlen(kitapAd)!= 0 && strcmp(kitapAd,"USERS")!=0)
  {
        struct KITAP* sonaEklenecek = kitapOlustur(kitapAd);
          if (kitapHead == NULL)
                {
                  kitapHead = sonaEklenecek;
                }
                else
                {
                        struct KITAP* tmp = kitapHead;
                        while(tmp->next != NULL)
                        {
                                tmp=tmp->next;
                        }
                        tmp->next=sonaEklenecek;
```

```
                }
                printf("%s isimli Kitap eklendi\n");
        }
}

void kitaplariYaz()//Kitaplar linkli listesinde bulunan tüm kitapları ekrana yazdırmak için kullanılır.
{
        int j = 0;
        struct KITAP *tmp = kitapHead;
        printf("\n====================\n");
        printf("KITAPLAR:\n");
        while(tmp != NULL)
        {
                printf("Ad:%s\n",tmp->ad);
                tmp = tmp->next;
        }
        printf("\n====================\n");
}

struct KITAP* kitapGetir(int siraNo)
{
        /*
        inputlar:  kitap sıra nosu
        Açıklama:          Sıra numarası girilen kitabın adresini döndürür.
        */
        int i=0;
        struct KITAP *tmpKitap=kitapHead;
        for(i=0;i<siraNo;i++)
        {
                tmpKitap=tmpKitap->next;
        }

        return tmpKitap;
}
```