

## **Six Degrees of Kevin Bacon**

Yıldız Teknik Üniversitesi Veri Yapıları ve Algoritmaları Burak Boz 18011706

### Nasıl Çalışır;

Program başlatıldığında önce kullanıcıya kullanım kılavuzu gösterilir ve adımların nasıl çalıştığı anlatılır. Kullanıcı herhangi bir tuşa basarak kullanım kılavuzunu atladıktan sonra program otomatik olarak input dosyasından verileri taramaya başlar. Filmler ve oyuncuları input dosyasında satır satır ve "/" operatörü kullanarak ayrılmıştır. İlk "/" operatörüne gelene kadar filmin adı, ardından oyuncular Soyad, Ad/ şeklinde ayrılmıştır. Bu aşamada filmler okunur ve ilgili düğümler oluşturulur. Düğüm içerisinde value, title, level ve parent değerleri bulunmaktadır. Filmler title içerisinde tutulur ve level larına varsayılan olarak -1 değeri atılır.

Tüm filmler okunduktan sonra 2. Aşama olan ana menüye geçilir. Burada kullanıcıya 3 seçim hakkı tanınmaktadır.

1 numaralı seçim herhangi bir oyuncu için Kevin Bacon derecesini hesaplamak için kullanılır.

Aranacak kişinin ismi Soyad, Ad şeklinde girildikten sonra bu kişi döngü içerisine konularak aranır ve sıralamadaki yeri tespit edilir. Daha sonra bağlantı olup olmadığı kontrol edilir. Eğer bağlantı var ise bağlantıyı sağlayan oyuncularla birlikte ekrana yazdırılır.

Bu işlem yapılırken aranan oyuncu ek bir tabloya kaydedilir ve 2. Seçim için bu tablo kullanılır.

2 numaralı seçim seçildiğinde girilen oyuncunun ismi bu tabloda aranır ve bulunursa sonuç hızlı bir şekilde ekrana yazdırılır. Eğer aranan oyuncu için daha önce bir hesaplama yapılmadıysa ekrana bu oyuncu için daha önceden hesaplama yapılmadığı uyarısı verilir.

3. Seçim çalıştırıldığında tüm oyuncular için Kevin Bacon bağlantı derecesi hesaplanır ve tüm oyuncular için tek tek yazılır. İşlem tamamlandığında her bir seviyede kaç adet oyuncu olduğu ekranda gösterilir.

## Ekran Çıktıları ve Açıklamaları;

#### input dosyası: input-1.txt

Her Kevin Bacon derecesinde kaç adet oyuncu bulunduğu aşağıdaki görseldedir.

```
Kevin Bacon Derecesi:1

Walker, Michael Dean
Kevin Bacon Derecesi:1

Wallace, Laurie
Kevin Bacon Derecesi:1

White, Rebecca
Kevin Bacon Derecesi:1

Wilson, Keith
Kevin Bacon Derecesi:1

Wilson, Keith
Kevin Bacon Derecesi:1

Wilson, Keith
Kevin Bacon Derecesi:1

Yßtez, Eduardo
Kevin Bacon Derecesi:1

Kevin Bacon derecesi 1 olan oyuncu sayisi: 178
Kevin Bacon derecesi 2 olan oyuncu sayisi: 234
Kevin Bacon derecesi 3 olan oyuncu sayisi: 120
Kevin Bacon derecesi 4 olan oyuncu sayisi: 120
Kevin Bacon derecesi 5 olan oyuncu sayisi: 125
Kevin Bacon derecesi 6 olan oyuncu sayisi: 125
Kevin Bacon derecesi 7 olan oyuncu sayisi: 125
Kevin Bacon derecesi 8 olan oyuncu sayisi: 184
Kevin Bacon derecesi 8 olan oyuncu sayisi: 83
Kevin Bacon derecesi 9 olan oyuncu sayisi: 84
Kevin Bacon derecesi 9 olan oyuncu sayisi: 293

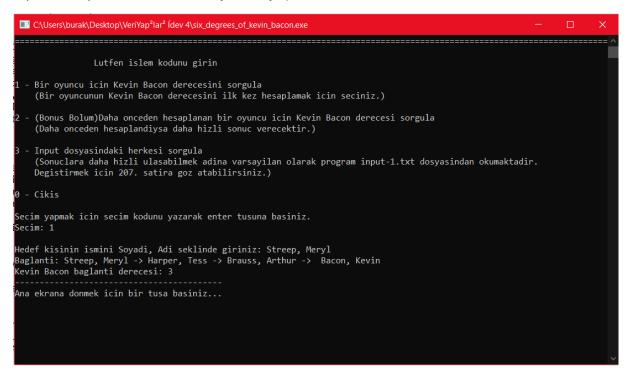
Ana ekrana donmek icin bir tusa basiniz...
```

#### İnput dosyası: input-2.txt

Her Kevin Bacon derecesinde kaç adet oyuncu bulunduğu aşağıdaki görseldedir.

İnput-3.txt dosyası'nın okuma süresi çok uzun olduğu için tüm seviyelerdeki Kevin Bacon derecelerinin ekran görüntüsü alınamamıştır.

#### İnput 3 dosyasından alınan **Meryl Streep** için Kevin Bacon derecesi:



#### İnput 3 dosyasından alınan **Nicolas Cage** için Kevin Bacon derecesi:

```
Lutfen islem kodunu girin

1 - Bir oyuncu icin Kevin Bacon derecesini sorgula
(Bir oyuncunun Kevin Bacon derecesini ilk kez hesaplamak icin seciniz.)

2 - (Bonus Bolum)Daha onceden hesaplanan bir oyuncu icin Kevin Bacon derecesi sorgula
(Daha onceden hesaplandiysa daha hizli sonuc verecektir.)

3 - Input dosyasindaki herkesi sorgula
(Sonuclara daha hizli ulasabilmek adina varsayilan olarak program input-1.txt dosyasindan okumaktadir.
Degistirmek icin 207. satira goz atabilirsiniz.)

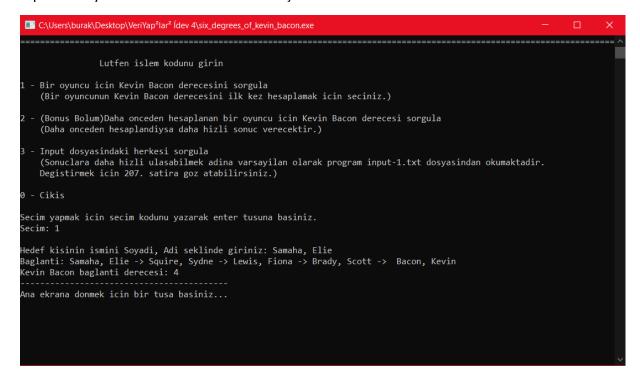
6 - Cikis

Secim yapmak icin secim kodunu yazarak enter tusuna basiniz.
Secim: Cage, Nicolas

Hedef kisinin ismini Soyadi, Adi seklinde giriniz: Baglanti: Cage, Nicolas -> Abercrombie, Ian -> Brady, Scott -> Bacon
(Kevin Bacon baglanti derecesi: 3

Ana ekrana donmek icin bir tusa basiniz...
```

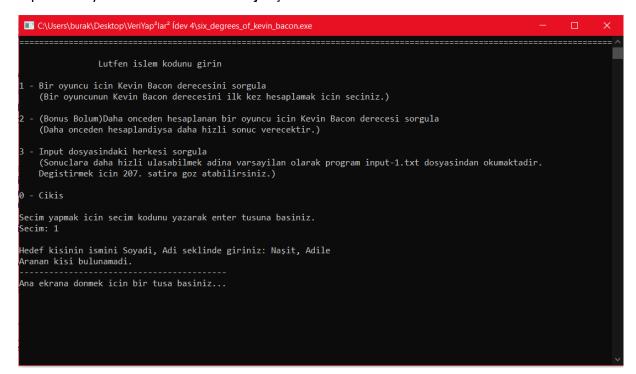
#### İnput 3 dosyasından alınan Elie Samaha için Kevin Bacon derecesi:



#### İnput 3 dosyasından alınan **Dakota Fanning** için Kevin Bacon derecesi:

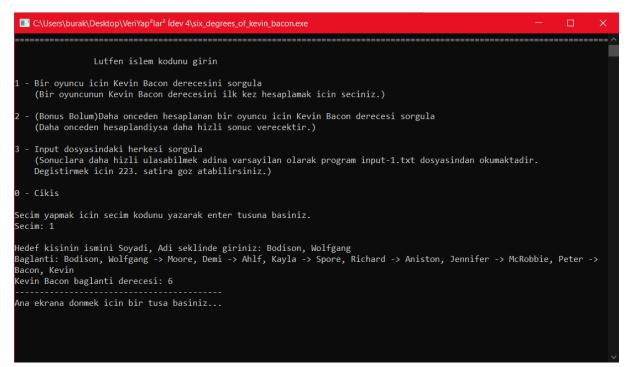
```
C:\Users\burak\Desktop\VeriYap²lar² Ídev 4\six_degrees_of_kevin_bacon.exe
 Lutfen islem kodunu girin
 - Bir oyuncu icin Kevin Bacon derecesini sorgula
   (Bir oyuncunun Kevin Bacon derecesini ilk kez hesaplamak icin seciniz.)
 - (Bonus Bolum)Daha onceden hesaplanan bir oyuncu icin Kevin Bacon derecesi sorgula
   (Daha onceden hesaplandiysa daha hizli sonuc verecektir.)
   Input dosyasindaki herkesi sorgula
   (Sonuclara daha hizli ulasabilmek adina varsayilan olarak program input-1.txt dosyasindan okumaktadir.
Degistirmek icin 207. satira goz atabilirsiniz.)
0 - Cikis
Secim yapmak icin secim kodunu yazarak enter tusuna basiniz.
Secim: Fanning, Dakota
Hedef kisinin ismini Soyadi, Adi seklinde giriniz: Baglanti: Fanning, Dakota -> Abatemarco, Tony -> Keaton, Diane -> Bea
tty, Warren -> Bacon, Kevin
Kevin Bacon baglanti derecesi: 4
Ana ekrana donmek icin bir tusa basiniz...
```

#### İnput 3 dosyasından alınan Adile Naşit için Kevin Bacon derecesi:



# (Bonus Bölüm) İnput-1.txt Dosyasıdan alınan, önceden aranmış oyuncunun tablodan yazdırılması;

Seçim 1 çalıştırılarak Wolfgang Badison ilk kez arandığında bağlantı derecesi bulunmuş ve tabloya kaydedilmiştir.

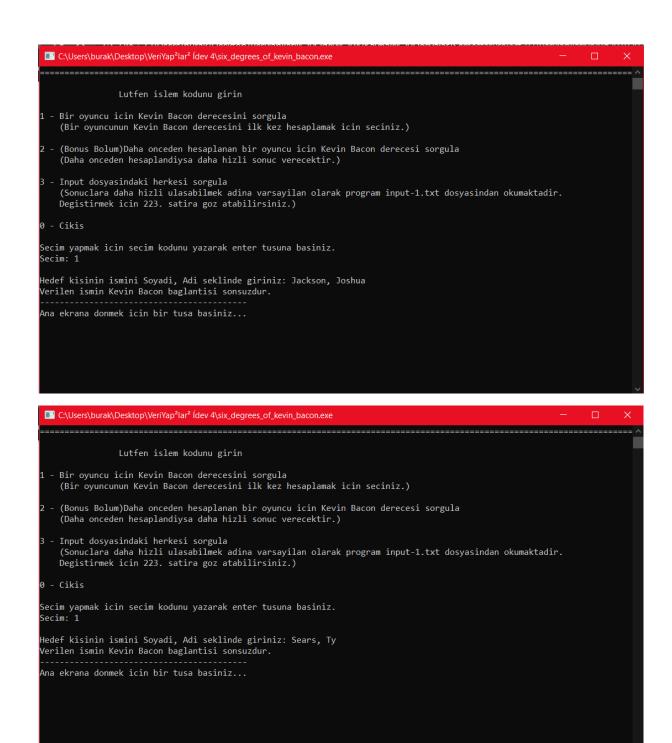


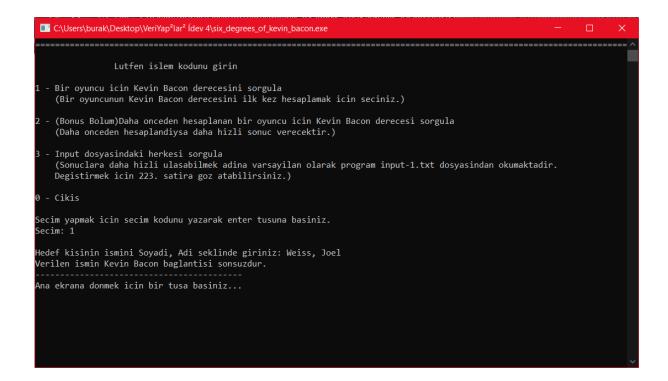
Seçim 2 çalıştırılarak Wolfgang Badison 2. Kez arandığında ise kaydedilen tablodan veriyi daha hızlı şekilde ekrana yazdırmıştır.



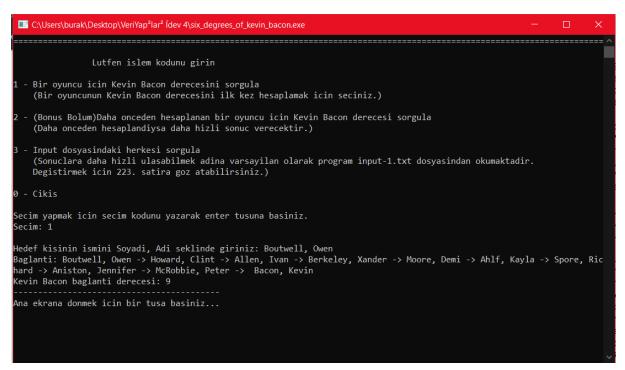
## İnput-1.txt Dosyasından alınan ve bağlantısı sonsuz olan 4 kişi;

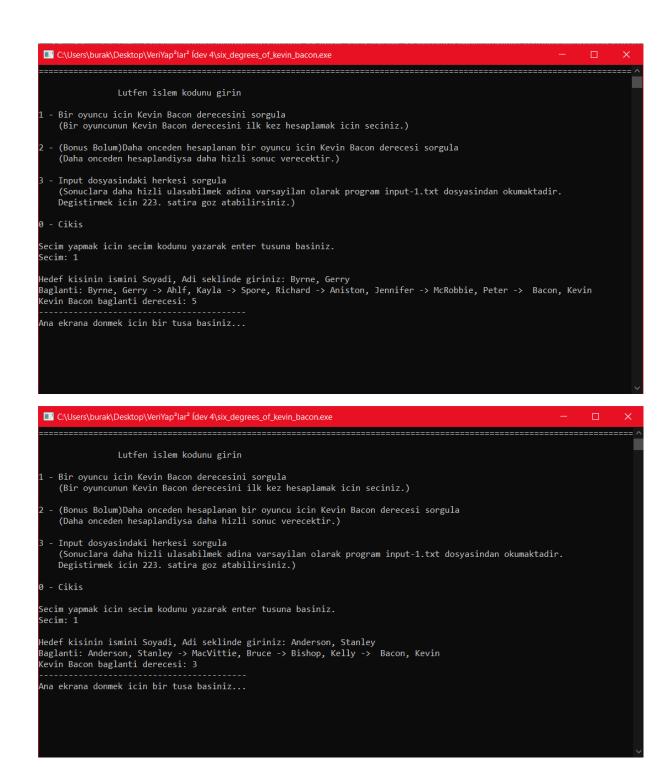






## İnput-1.txt Dosyasından alınan ve bağlantısı olan olan 4 kişi;





```
Lutfen islem kodunu girin

1 - Bir oyuncu icin Kevin Bacon derecesini sorgula
(Bir oyuncunun Kevin Bacon derecesini ilk kez hesaplamak icin seciniz.)

2 - (Bonus Bolum)Daha onceden hesaplanan bir oyuncu icin Kevin Bacon derecesi sorgula
(Daha onceden hesaplandiysa daha hizli sonuc verecektir.)

3 - Input dosyasindaki herkesi sorgula
(Sonuclara daha hizli ulasabilmek adina varsayilan olarak program input-1.txt dosyasindan okumaktadir.
Degistirmek icin 223. satira goz atabilirsiniz.)

6 - Cikis

Secim yapmak icin secim kodunu yazarak enter tusuna basiniz.
Secim: 1

Hedef kisinin ismini Soyadi, Adi seklinde giriniz: Avon, Wendy Lee
Baglanti: Avon, Wendy Lee -> Baker, Dylan -> Candy, John -> Alciatore, Henri -> Johnston, J.J. -> Bacon, Kevin Kevin Bacon baglanti derecesi: 5

Ana ekrana donmek icin bir tusa basiniz...
```

## Programın Kaynak Kodları;

/\*

Name: Burak Boz

Student no: 18011706

E-Mail: burakbozb1@gmail.com

Operating System: Windows 10 64-bit

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int found=0;//Bayrak olarak kullanılacaktır

int anlikHash=1;//Arama yaparken index değeri olarak kullanılacaktır.

int queueSize=1;//BFS araması yaparken veriler kuyruğa dizilecektir. Kuyruğun boyutu için kullanılacaktır.

int visitedSize=1;//Grafta ziyaret edilen düğümlerin sayısını tutacaktır.

```
int aramaLeveli=0;//Arama seviyesi için kullanılacaktır.
```

```
typedef struct Node{
        int value;
        char title[256];//Film ya da oyuncu ismi tutulacaktır.
        int level;//Seviyeyi tutacaktır. -2 sonsuz bağlantıyı, -1 ise film olduğunu gösterecektir. Diğer durumlar
ise bağlantı seviyesini temlsil edecektir.
        int parent;//Grafın düğümleri arasındaki bağlantı için kullanılacaktır.
}Node;
int baglantiVarMi(int anlikHash,int arananHash,int** adjacencyList,Node* nodeArray,int aramaLeveli){//Aranan
oyuncunun bağlantısının olup olmadığı kontrol ediliyor.
        //Bağlantı var ise 1 döndürecektir. Yoksa 0 döndürecektir.
        int k=0;
        //printf("%d ile %d baglantili mi?",anlikHash,arananHash);
        while((adjacencyList[anlikHash][k]!=arananHash)&&(adjacencyList[anlikHash][k]!=0)){
                 k++;
        }
        if(adjacencyList[anlikHash][k]!=arananHash){
                 //printf(" | Hayir.\n");
                 return 0;
        }else{
                 nodeArray[arananHash].parent=anlikHash;
                 //printf(" | Evet.\n");
                 found=1;
                 //printf("\tBagli Film:%d\n",nodeArray[k].value);
                 return 1;
        }
}
```

int hashAra(int anlikHash,int arananHash,int\*\* adjacencyList,int aramaLeveli,Node\* nodeArray,int\* queue,int\* visited,int queueMax){

```
//Bu fonksiyon aranan kişinin hash değerini döndürecek.
        //Ayrıca aranan kişinin daha önce aranıp aranmadığını kontrol edecek.
        //Eğer kişi daha önceden zaten aranmışsa kaydedilen tablodan yazacak ve daha hızlı sonuç verecek.
        int i,j;
        int k=0;
        int queueIndex=1;
        int flag=0;
        for(i=0;i<visitedSize;i++){</pre>
                 if(visited[i]==queue[0]){
                          flag=1;
                 }
        }
        if(flag==0){
                 visitedSize++;
                 visited=(int*) realloc(visited,(visitedSize)*sizeof(int));
                 visited[visitedSize-1]=queue[0];
        }
        flag=0;
        found=baglantiVarMi(anlikHash,arananHash,adjacencyList,nodeArray,aramaLeveli);
                                                                                                //Aranan hash
değerindeki kişinin bağlantısının olup olmadığını kontrol ediyoruz.
        if(found==1){//Eğer kişi daha önceden ziyaret edilip bulunduysa hızlıca sonuç dönecek
                 nodeArray[arananHash].parent=1;
                 return arananHash;
        }else{//Eğer kişi ilk defa aranıyorsa gerekli kontroller yapılacak ve bağlantı durumu kontrol edilecek
                 while(adjacencyList[anlikHash][k]!=0){
                          k++;
                 }
```

```
for(i=0;i< k;i++){
                                                                                                                     //printf("Komsular %d || ",anlikHash);
                             baglanti Var Mi (node Array [adjacency List [anlik Hash][i]]. value, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, aranan Hash, adjacency List, node Array, adjacency List, node Array, adjacency List, node Array, adjacency List, node Array, adjacency List, node Array, adjacency List, node Array, node Array, adjacency List, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node Array, node 
maLeveli);
                                                                                                                      if(found==1){
                                                                                                                                                   free(visited);
                                                                                                                                                   free(queue);
                                                                                                                                                   nodeArray[adjacencyList[anlikHash][i]].parent=anlikHash;
                                                                                                                                                    nodeArray[adjacencyList[anlikHash][i]].level=aramaLeveli+1;
                                                                                                                                                    nodeArray[arananHash].level=aramaLeveli+2;
                                                                                                                                                   return arananHash;
                                                                                                                      }else{
                                                                                                                                                   for(j=0;j<visitedSize;j++){</pre>
                                                                                                                                                                                if(visited[j]==adjacencyList[anlikHash][i]){
                                                                                                                                                                                                              flag=1;
                                                                                                                                                                                }
                                                                                                                                                   }
                                                                                                                                                   for(j=0;j<queueSize;j++){</pre>
                                                                                                                                                                                if(queue[j]==adjacencyList[anlikHash][i]){
                                                                                                                                                                                                              flag=1;
                                                                                                                                                                                }
                                                                                                                                                   }
                                                                                                                                                   if(flag==0){
                                                                                                                                                                                queueSize++;
                                                                                                                                                                                queue= (int*) realloc(queue,(queueSize+1)*sizeof(int));
                                                                                                                                                                                queue[queueSize-
1]=nodeArray[adjacencyList[anlikHash][i]].value;
                                                                                                                                                                                 nodeArray[queue[queueSize-1]].parent=anlikHash;
                                                                                                                                                                                 nodeArray[queue[queueSize-1]].level=aramaLeveli+1;
```

```
}
                            /*
                           if (node Array [adjacency List [anlik Hash][i]].parent == 0) \{\\
                                     node Array [adjacency List [anlik Hash][i]]. parent = anlik Hash; \\
                           }
                            */
                  }
                  flag=0;
         }
}
flag=0;
for(i=1;i<queueSize;i++){</pre>
         nodeArray[queue[i]].parent=queue[0];
}
*/
if(queue[0]==queueMax){
         aramaLeveli++;
         queueMax=queue[queueSize-1];
}
queue=(int*) realloc(queue,(queueSize+1)*sizeof(int));
for(i=0;i<queueSize-1;i++){</pre>
         queue[i]=queue[i+1];
}
if (queue Size == 0) \{\\
         return -1;
}else{
         queue[i]=0;
         queueSize--;
```

```
}
```

```
hashAra(queue[0],arananHash,adjacencyList,aramaLeveli,nodeArray,queue,visited,queueMax);
       }
int main() {
       //Kullanim klavuzu**
       =======\n\n");
       printf("\t\tCreated By Burak BOZ - 18011706 - 16.05.2020\n");
       printf("\n*** Six Degrees of Kevin Bacon Kullanim Klavuzu ***\n");
       printf("1* Programi calistirmak icin, programin exe dosyasi ile ayni dizinde input-1.txt isimli text
dosyasi bulunmalidir.\n");
       printf(" Herhangi bir tusa basarak kullanim klavuzunu gectikten sonra program otomatik olarak
filmleri taramaya baslayacaktir.\n\n");
       printf("2* Ozel bir oyuncunun derecesini hesaplamak icin 1. ve 2. secimler kullanılabilir. Eger bir ismi ilk
kez arayacaksaniz \n 1. secimi kullanınız.\n");
       printf(" Eger ilk kez arayacaginiz bir oyuncuyu 2. adimda aramaya kalkarsaniz daha onceden
hesaplanmadigi \n uyarisini alacaksiniz.\n");
       printf(" 1. secimde aradiginiz kisi bir tablo uzerine kaydedilecek ve daha sonra tekrar aramak
istediginizde \n 2. secimi kullanarak daha hizli sonuc alacaksiniz.\n");
       printf(" 1. ve 2. secimlerde aramak istediginiz oyuncunun adini ve soyadini 'Soyad, Ad' Seklinde
girmelisiniz. \n Ornek: Plummer, Amanda\n\n");
       printf("3* Eger input dosyasindaki tum isimlerin Kevin Bacon derecelerini hesaplamak ve gormek \n
istiyosaniz 3. adimi calistirarak tum oyuncular icin derecelerin hesaplanmasini saglamalisiniz.\n");
       printf(" Bu adimi calistirarak tum oyunculari hesapladiktan sonra tum oyuncularin Kevin Bacon
derecelerini \n bireysel olarak goreceksiniz. Ardindan tablo seklinde her seviyede kac adet oyuncu\n
oldugunu goreceksiniz.\n");
       =========\n");
       printf("\nKullanim klavuzunu gecmek icin herhangi bir tusa basiniz.");
       getch();
       //Kullanim klavuzu sonu
```

```
int secim=-1;//Kullanıcının seçimi bu değişken içerisinde saklanacak.
        int dKontrol=0; //3. seçimin dogru calismasi icin yapilacak kontrol
        char buffer[2048];//Satır satır okuma yapılırken bu char dizisi kullanılacak
        int hamle[10];
        char* asd;
        int value=1;
        char* token;
                          //Satırdaki elemanlar "/" operatörüne göre bölünecek ve arada alınan değerler bu
pointer ile saklanacak.
        Node* nodeArray =(Node *) calloc (1,sizeof(Node)); //Düğümlerimizi bu dizi içerisinde saklayacağız.
        int flag=0;
                          //Bazı durumların kontrolünde bayrak olarak kullanılacak.
        //Döngülerimiz için kullanılacak değişkenler
        int i;
        int j;
        int k;
        int I;
        int m=0;
        int n;
                                  //Geçici değer tutacaktır.
        int tmpValue;
        int arananHash;
                                  //Aranan oyuncunun hash değeri bu değişkende saklanacak.
        int arananYedek; //Aranan hash değeri üzerinde işlem yapacağımız için yedek olarak bu değişkeni
kullanacağız.
        int bulunanHash;//Fonksiyonda arama yaptıktan sonra bu değişkende sonuç tutulacak.
        int headHash=1;
                                  //İlk düğümü tutacak
        int hamleSayisi=0;
        int** adjacencyList;//Bağlantı listesinin başlangıç adresini tutacaktır.
        int adjacencyMax=0;//Bağlantı listesinin maximum eleman sayısını tutacaktır.
        int tmpOldMax=0;//Maximum değer güncellenirken geçici değer tutmak için kullanılacaktır.
        int* valueList=(int *) calloc(1,sizeof(int)); //Değer listesi için bellekte yer açılıyor.
        int* queue=(int *)malloc(2*sizeof(int));
                                                            //BFS Araması yapılırken kuyruk mantığı
kullanılacak.
```

```
int* visited=(int*)malloc(2*sizeof(int));
                                                              //Ziyaret edilenler tabloda tutulacak. Bu veriler
için bellekte yer açılıyor.
         int frekans[10]={0};
                                   //Kevin Bacon gruplarını tutacak olan dizi. Örnek: KB sayısı 5(i) olanların
sayısı = 15(frekans[i])
         int sonsuzKisi;
                                             //Kevin Bacon sayısı sonsuz olanların sayısını tutacak değişken
                          //Frekansı O olan tek kişi Kevin Bacon'ın kendisidir. Bu sebeple ilk değere default
         frekans[0]=1;
olarak 1 atanıyor.
         visited[0]=1;
                          //İlkleme yapılıyor.
         queue[0]=1;
                                   //İlkleme yapılıyor.
         adjacencyList = (int**) calloc ((adjacencyMax+2),sizeof(int*));
                                                                                //Bellekte kullanılacak olan yer
ayarlamaları yapılıyor.
         for(i=0;i<(adjacencyMax+1);i++){</pre>
                  adjacencyList[i]=(int*) calloc(1,sizeof(int));
        }
         adjacencyList[1]= (int*) realloc(adjacencyList[1],2*sizeof(int));
         FILE *fp = fopen("input-1.txt","r"); //İnput dosyası açılıyor.
         if (fp!= NULL){ //Eğer dosya bulunduysa
         while(fgets ( buffer, sizeof buffer, fp) != NULL) {//Satır satır okuma işlemi başlıyor
                  k=0;
                           token=strtok(buffer,"/"); // "/" karakterine göre bölünüyor ve film ismi okunuyor.
                           printf("Film ismi: %s\n",token);
                           token=strtok(NULL,"/");
                           while(token != NULL){ //Veri problemsiz şekilde alındıysa döngü çalışmaya
başlayacak. Bu döngü satır boyunca çalışacak, filmin ve oyuncuların tespitinde kullanılacak.
                                    if(token[strlen(token)-1]=='\n'){ //Token içerisine film ya da oyuncunun
```

token[strlen(token)-1]='\000';

alınmasında kullanılıyor.

```
}
                                   if(strcmp(token,"\n")!=0){
                                           for(i=1;i<value;i++){
                                                    if(strcmp(token,nodeArray[i].title)==0){
                                                             flag=1;
                                                             tmpValue=nodeArray[i].value;
                                                    }
                                           }
                                           /*
                                           Flag kontrolü
                                           Eğer flag 0 ise film daha önce eklenmemiştir. Film için yeni düğüm
oluşturuluyor.
                                           Eğer flag 0 değilse film eklenmiştir ve oyuncuların filme eklenmesi
işlemi gerçekleştirilir.
                                           */
                                           if(flag==0){//Yeni düğüm oluşturuluyor ve title alanına filmin adı
ekleniyor
                                                    struct Node newNode;
                                                    newNode.value=value;
                                                    strcpy(newNode.title,token);
                                                    newNode.level=-1;//Oluşturulan filmin başlangıçta level
değeri varsayılan olarak -1 olarak ayarlanıyor.
                                                    nodeArray =(Node *) realloc
(nodeArray,(value+1)*sizeof(Node));
                                           //NodeArray in boyutu güncelleniyor ve yeni düğüm için yer
açılıyor.
                                                    nodeArray[value]=newNode;
                                                                                       //Yeni açılan alana,
oluşturulan yeni düğüm ekleniyor.
                                                    valueList[k]=value;
                                                    valueList= (int*) realloc (valueList,(k+2)*sizeof(int));
                                                    k++;
```

```
value++;
```

```
}else{//Eğer zaten film eklenmişse filmin oyuncusu düğüme
ekleniyor.
                                                     valueList= (int*) realloc (valueList,(k+2)*sizeof(int));
                                                     valueList[k]=tmpValue;
                                                     k++;
                                                     tmpValue=nodeArray[i].value;
                                                     flag=0;
                                            }
                                   }
                                   token=strtok(NULL,"/");
                          }
                          if(value>adjacencyMax){ //Bağlantı Listesi için bellekte ayrılan yer yetersiz kalırsa,
yeni alan açılıyor ve gerekli değişkenler güncelleniyor.
                                                     tmpOldMax=adjacencyMax;
                                                     adjacencyMax=value;
                                                     adjacencyList = (int**) realloc
(adjacencyList,(adjacencyMax+1)*sizeof(int*));
                                                     for(i=tmpOldMax;i<adjacencyMax+1;i++){</pre>
                                                              adjacencyList[i]= (int*)calloc(1,sizeof(int));
                                                     }
                                            }
                          flag=0;
                          j=0;
                          I=0;
                          //Oyuncuların yakınlık dereceleri hesaplanacak
                          for(j=0;j< k-1;j++){
                                   for(|=j+1;|< k;|++){}
                                            n=0;
```

```
while ((adjacencyList[valueList[j]][n] > 0) \& (adjacencyList[valueList[j]][n]! = valueList[l])) \{ adjacencyList[valueList[j]][n]! = valueList[l]) \} \} \\
                                                                 n++;
                                                      }
                                                      adjacencyList[valueList[j]]= (int*)
realloc(adjacencyList[valueList[j]],(n+2)*sizeof(int));
                                                      adjacencyList[valueList[j]][n]=valueList[l];
                                                      adjacencyList[valueList[j]][n+1]=0;
                                                      n=0;
          while ((adjacencyList[valueList[l]][n] > 0) \& (adjacencyList[valueList[l]][n]! = valueList[j])) \{ (adjacencyList[valueList[l]][n]! = valueList[j]) \} \} \} 
                                                                 n++;
                                                      }
                                                      if((valueList[j]==304)\&\&(valueList[l]==102))\{\\
                                                                 printf(asd);
                                                      }
                                                      adjacencyList[valueList[l]]= (int*)
realloc(adjacencyList[valueList[I]],(n+2)*sizeof(int));
                                                      adjacencyList[valueList[l]][n]=valueList[j];
                                                      adjacencyList[valueList[l]][n+1]=0;
                                           }
                                }
                                 free(valueList);
                                 valueList=(int *) calloc(1,sizeof(int));
                                 m++;
                                 }
                     }
                      //Input dosyasındaki tüm filmler okundu.
```

```
printf("\n\nFilmlerin yuklenmesi tamamlandi.\n");
               printf("\n-----");
               printf("\nDevam etmek icin bir tusa basiniz...");
               getch();
       char aranan[64];
       int sonsuzFlag=0;
       while(secim){ //Kullanıcıya yapmak istediği işlem soruluyor. Verdiği cevap seçim değişkeni
içerisinde saklanacak. Ve switch case yapısında seçimine göre işlem yapılacak.
               sonsuzFlag=0;
               system("cls");
       =======\n\n");
               printf("\t\tLutfen islem kodunu girin\n\n");
               printf("1 - Bir oyuncu icin Kevin Bacon derecesini sorgula \n (Bir oyuncunun Kevin Bacon
derecesini ilk kez hesaplamak icin seciniz.)\n\n"); //Herhangi bi oyuncunun kevin bacon derecesi ilk kez
hesaplanacaksa bu seçime girilmelidir.
               printf("2 - (Bonus Bolum)Daha onceden hesaplanan bir oyuncu icin Kevin Bacon derecesi
sorgula \n (Daha onceden hesaplandiysa daha hizli sonuc verecektir.)\n\n"); //Oyuncunun KevinBacon
derecesi daha önceden hesaplandıysa ve hızlıca sonuç almak isteniyorsa bu seçime girilmelidir.
               printf("3 - Input dosyasindaki herkesi sorgula\n (Sonuclara daha hizli ulasabilmek adina
varsayilan olarak program input-1.txt dosyasindan okumaktadir.\n Degistirmek icin 223. satira goz
atabilirsiniz.)\n\n");
                       //İnput dosyası içerisideki tüm oyuncular için sonuç görmek istenirse bu seçime
girilmelidir.
               printf("0 - Cikis\n\n");
                                      //Programı sonlandırmak için bu seçime girilmeilidir.
               printf("Secim yapmak icin secim kodunu yazarak enter tusuna basiniz.\n");
               printf("Secim: ");
               scanf("%d%*c",&secim);
               printf("\n");
               switch(secim){
                       case 1: //Eğer seçim 1 ise, girilecek oyuncu ile kevin bacon arasındaki bağlantı
sorgulanacak.
```

```
fgets(aranan,100,stdin);
                                 aranan[strlen(aranan)-1]='\0';
                                 flag=0;
                                 while((strcmp(nodeArray[arananHash].title,aranan)!=0)&&(flag==0)){
                                          arananHash++;
                                          if(arananHash==value){
                                                  printf("Aranan kisi bulunamadi.");
                                                  flag=1;
                                         }
                                 }
                                 if(flag==0){
                                          arananHash=nodeArray[arananHash].value;
                                          arananYedek=arananHash;
                                                                                   //ArananHash işlemde
kullanılacak. Değeri kaybetmemek için arananYedek değişkeninde yedekliyoruz.
                                          hamleSayisi=0;
                                          if(strcmp(nodeArray[arananHash].title,"Bacon, Kevin")==0){//Eğer
Kevin Bacon'un kendisi aranıyorsa 0 değeri ekrana yazdırılacak.
                                                  printf("Kevin Bacon'un Kevin Bacon degeri 0.");
                                          }else{ //Kevin Bacon'un kendisi aranmıyorsa hash değeri
hesaplanacak.
        bulunanHash=hashAra(queue[0],arananHash,adjacencyList,0,nodeArray,queue,visited,1);
                                                  sonsuzFlag=bulunanHash;
                                                  bulunanHash=arananHash;
                                                  if(sonsuzFlag==-1){//Aranan kişinin bağlantı derecesi -1 ise
sonsuz olarak ekrana yazdırılacak.
```

printf("Hedef kisinin ismini Soyadi, Adi seklinde giriniz: ");

```
printf("Verilen ismin Kevin Bacon baglantisi
sonsuzdur.");
                                                           nodeArray[arananHash].level=-2;
                                                  }else{
                                                           printf("Baglanti: ");
                                                           while(nodeArray[bulunanHash].value!=1){
                                                                   printf("%s ->
",nodeArray[bulunanHash].title);
                                                                   hamleSayisi++;
        bulunanHash=nodeArray[bulunanHash].parent;
                                                                   if(nodeArray[bulunanHash].parent==1){
                                                                            hamleSayisi++;
                                                                            printf("%s ->
",nodeArray[bulunanHash].title);
                                                                            bulunanHash=1;
                                                                   }
                                                           }
                                                           printf(" Bacon, Kevin");
                                                           if(nodeArray[arananYedek].level==-1){
                                                                   nodeArray[arananYedek].level=1;
                                                           }
                                                           printf("\nKevin Bacon baglanti derecesi:
%d",nodeArray[arananYedek].level);
                                                  }
                                         }
                                 }
                                 visitedSize=1;
                                 queueSize=1;
```

```
visited[0]=1;
                                 queue=(int *)calloc(2,sizeof(int));
                                 queue[0]=1;
                                  arananHash=1;
                                  printf("\n----");
                                  printf("\nAna ekrana donmek icin bir tusa basiniz...");
                                 getch();
                                  break;
                         case 2://(Bonus bölüm) Eğer girilen oyuncu daha önce hesaplandıysa daha hizli sonuc
verecek. Eğer oyuncu ilk kez aranıyorsa daha önce hesaplanmadığını söyleyecek.
                                 printf("Hedef kisinin ismini Soyadi, Adi seklinde giriniz: ");
                                 fgets(aranan,100,stdin);
                                 aranan[strlen(aranan)-1]='\0';
                                 while((strcmp(nodeArray[arananHash].title,aranan)!=0)&&(flag==0)){
                                          arananHash++;
                                          if(arananHash==value){
                                                  printf("Aranan kisi bulunamadi.");
                                                  flag=1;
                                          }
                                 }
                                 if(flag==0){
                                          if(nodeArray[arananHash].level==-1){
                                                   printf("Aranan kisinin Kevin Bacon numarasi henuz
hesaplanmamis.");
                                          }else if(nodeArray[arananHash].level==-2){
                                                  printf("Aranan kisinin Kevin Bacon numarasi sonsuz.");
                                          }else{
                                                   printf("Aranan kisinin Kevin Bacon derecesi sistemde
kayitlidir.\nKevin Bacon baglanti derecesi: %d",nodeArray[arananHash].level);
```

visited=(int\*)calloc(2,sizeof(int));

```
}
                                  printf("\nAna ekrana donmek icin bir tusa basiniz...");
                                  getch();
                                  break;
                         case 3://Tüm oyuncular için Kevin Bacon derecelerini hesaplayacaktır.
                                  dKontrol++;
                                  int valueYedek,sayacB=0;
                                  valueYedek=value;
                                  printf("Input dosyasindaki butun isimler araniyor...\n");
                                  for(i=2;i<=value;i++){
                                          if(nodeArray[i].level==-1){
                                                   printf("%s sorgulaniyor...\n",nodeArray[i].title);
                                                   arananHash=nodeArray[i].value;
                                                   arananYedek=arananHash;
                                                   hamleSayisi=0;
                                                   if(strcmp(nodeArray[arananHash].title,"Bacon,
Kevin")==0){
                                                            //printf("Kevin Bacon'un Kevin Bacon degeri 0.");
                                                   }else{
        bulunanHash=hashAra(queue[0],arananHash,adjacencyList,0,nodeArray,queue,visited,1);
                                                            sonsuzFlag=bulunanHash;
                                                            if(sonsuzFlag==-1){
```

}

```
baglantisi sonsuzdur.\n");
                                                                    nodeArray[arananHash].level=-2;
                                                           }else{
                                                                    bulunanHash=arananHash;
        while(nodeArray[bulunanHash].value!=1){
                                                                            hamleSayisi++;
        bulunanHash=nodeArray[bulunanHash].parent;
        if(nodeArray[bulunanHash].parent==1){
                                                                                     hamleSayisi++;
                                                                                     bulunanHash=1;
                                                                            }
                                                                    }
                                                                    if(nodeArray[arananYedek].level==-1){
        nodeArray[arananYedek].level=1;
                                                                    }
                                                                    //printf("%s kisinin Kevin Bacon
derecesi: \%d\n", node Array[bulunan Hash]. title, node Array[bulunan Hash]. level);
                                                           }
                                                  }
                                          }
                                          visitedSize=1;
                                          queueSize=1;
                                          visited=(int*)calloc(2,sizeof(int));
                                          visited[0]=1;
                                          queue=(int *)calloc(2,sizeof(int));
                                          queue[0]=1;
                                          arananHash=1;
```

//printf("Verilen ismin Kevin Bacon

```
}
                                  //Ekrana tablonun yazdırılması
                                  int seviye[20]={0};
                                  int sonsuzAdedi=0;
                                  int dSayac=0;
                                  for(sayacB=2;sayacB<valueYedek;sayacB++){</pre>
                                           if(nodeArray[sayacB].level==-2){
                                                    printf("%s \n\tKevin Bacon
Derecesi:sonsuz\n\n",nodeArray[sayacB].title);
                                                    //seviye[nodeArray[sayacB].level]++;
                                                    sonsuzAdedi++;
                                           }else if(nodeArray[sayacB].level==-1){
                                           }
                                           else if(nodeArray[sayacB].level==0)
                                           {
                                           }
                                           else
                                           {
                                                    printf("%s \n\tKevin Bacon
Derecesi:%d\n\n",nodeArray[sayacB].title,nodeArray[sayacB].level);
                                                    seviye[nodeArray[sayacB].level]++;
                                           }
                                  }
                                  for(dSayac=0;dSayac<20;dSayac++)</pre>
                                  {
                                           if(seviye[dSayac]!=0)
```

```
printf("Kevin Bacon derecesi %d olan oyuncu sayisi:
%d\n",dSayac,seviye[dSayac]);
                              }
                              printf("Kevin Bacon derecesi SONSUZ olan oyuncu sayisi:
%d\n",sonsuzAdedi);
                              printf("\n----");
                              printf("\nAna ekrana donmek icin bir tusa basiniz...");
                              getch();
                              break;
                       case 0:
                              exit(0);
                              break;
                       default:
                              printf("Gecersiz secim.");
                              printf("\n----");
                              printf("\nAna ekrana donmek icin bir tusa basiniz...");
                              getch();
                              break;
               }
       }
       return 0;
}
```