Dersin adı:Algoritma Analizi

Grup Numarası-2

Proje

Recep Furkan Koçyiğit

16011043

Ödev Konusu: Graf Üzerinde Arama İşlemi

Kaynak Kod:

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

#define TABLE_SIZE 250007
#define FILM_SIZE 14500
#define BUFFER_SIZE 3000

struct node{//film veya oyuncuya ait bilgileri tutar.
	char *data;//film veya oyuncunun ad bilgisini tutar.
	int index;//eklendiği yerin indis bilgisini tutar.
	struct node *next;
};
struct graph{//oyuncular ve filmlere ait graf
	struct node **artist;
	struct node **film;
};
typedef struct graph Graph;
struct stack{//oyunculardaki farkli kombinasyonlari tutmak icin stackler ard arda eklenerek tutulmustur.
	int len;//yiginin uzunlugu
	struct node *top;//yiginin en ustunu gosteren pointer
	struct stack *next;//bir sonraki stackin adresi
};
struct queue{//bfs yapılırken node ları gezmek icin kullanilmistir.
	struct stack *rear;
	struct stack *front;
};
void push(struct stack *s,char *data,int index){//verilen stackin en ustune verilen bilgilere ait node eklenir.
```

```
	struct node *newnode=(struct node *)malloc(sizeof(struct node));//stack e eleman eklemek icin
gecici eleman
	newnode->data=(char *)malloc(sizeof(char)*strlen(data));
	strcpy(newnode->data,data);
	newnode->next=NULL;
	newnode->index=index;
	if(s->top == NULL){
		s->top=newnode;
		s->len=1;
		return;
	}
	newnode->next=s->top;
	s->len=s->len+1;
	s->top=newnode;
}
void pop(struct stack *s){//stackin en ustundeki node u stackten siler.
	if(s->top == NULL){
		s->len=0;
		return;
	}
	struct node *temp=s->top;//stack ten eleman silmek icin gecici eleman
	s->top=s->top->next;
	s->len=s->len-1;
}
void print_stack(struct stack *s){//stackin icindeki nodelara ait bilgileri yazdirir.
	if(s->top == NULL){
		printf("Stack bos.\n");
		return;
	}
	struct node *iter=s->top;//stackin icini yazdirirken iterasyon amacli
	struct node *before;//stackin icini yazdirirken iterasyon amacli
```

```
	while(iter->next != NULL){
		printf("%s - ",iter->data);
		iter=iter->next;
		before=iter;
		iter=iter->next;
		printf("%s : ",iter->data);
		printf("%s\n",before->data);
	}
}
void print_queue(struct queue *q){
	struct stack *iter=q->front;//queue nun icindeki stackleri yazdirirken iterasyon amacli
	if(q->rear == NULL){
		printf("Queue bos.\n");
		return;
	}
	while(iter->next != NULL){
		print_stack(iter);
		iter=iter->next;
	}
	print_stack(iter);
}
void enqueue(struct queue *q,struct stack *s){//queue y yeni bir stack ekler
	struct stack *newstack=(struct stack *)malloc(sizeof(struct stack));////queue ya stack eklemek icin gecici eleman
	newstack->top=s->top;
	newstack->next=NULL;
	newstack->len=s->len;
	if(q->rear == NULL){
		q->front=newstack;
		q->rear=newstack;
		return;
```

```
	}
	q->rear->next=newstack;
	q->rear=newstack;
}
struct stack *dequeue(struct queue *q){//queue daki ilk eleman dondurur ve queue dan cikarir.
	if(q->rear == NULL){
		return;
	}
	struct stack *temp=q->front;
	q->front=q->front->next;
	if(q->front == NULL){
		q->rear=NULL;
	}
	return temp;
}
Graph *createGraph(){//grafi olusturmak icin yazilmis fonksiyon
	Graph *g=(Graph *)malloc(sizeof(Graph));//grafi olusturmak icin gecici olarak kullanilmis eleman
	g->artist=(struct node **)malloc(TABLE_SIZE * sizeof(struct node *));
	g->film=(struct node **)malloc(FILM_SIZE * sizeof(struct node *));
	int i;//iterasyon amacli kullanilmistir.
	for(i=0;i<TABLE_SIZE;i++){//oyunculari bos olarak ayarlama
		g->artist[i]=NULL;
	}
	for(i=0;i<FILM_SIZE;i++){//filmleri bos olarak ayarlama
		g->film[i]=NULL;
	}
	return g;
}
struct node *createNode(char *data){//verilen bilgiye gore node olusturur ve olusturulan node u dondurur.
```

```
    struct node *newNode=(struct node *)malloc(sizeof(struct node));//node olusturmak icin kullanilan gecici eleman

    newNode->data=(char *)malloc(sizeof(char)*strlen(data));

    strcpy(newNode->data,data);

    newNode->next = NULL;

    return newNode;

}

void addEdge(Graph *g,int index,char *artist){//verilen grafadaki verilen film indexi ile verilen oyuncu bilgisi arasinda kenar olusuturur.

        int dest=search(g,artist);//oyuncunun tabloda gelecegi index

        struct node *newNode;//verilen oyuncuyu grafa eklemek ve oyuncu ile film arasinda kenar olusturmak icin kullanilmistir.



        newNode=createNode(artist);//filme oyuncuyu ekleme

        newNode->index=dest;

    newNode->next=g->film[index]->next;

        g->film[index]->next=newNode;



        if(g->artist[dest] == NULL){//eger oyuncu ilk defa eklenecekse

                newNode=createNode(artist);

                newNode->index=dest;

                g->artist[dest]=newNode;

        }

        newNode=createNode(g->film[index]->data);//oyuncuya filmi ekleme

        newNode->index=index;

        newNode->next=g->artist[dest]->next;

        g->artist[dest]->next=newNode;



}

void printArtists(Graph *g){//graftaki oyunculari yazdirir

    int i;//iterasyon amacli kullanilmistir.
```

```
    for(i=0;i<TABLE_SIZE;i++){
        struct node *iter=g->artist[i];
        while(iter){
            printf("%s--",iter->data);
            iter=iter->next;
        }
        printf("\n");
    }
}
void printFilms(Graph *g){//graftaki filmleri yazdirir
    int i; //iterasyon amacli kullanilmistir.
    for(i=0;i<FILM_SIZE;i++){
        struct node *iter=g->film[i];
        while(iter){
            printf("%s--",iter->data);
            iter=iter->next;
        }
        printf("\n");
    }
}
void bfs(Graph *g,char *first,char *second,int f){//verilen grafta,verilen iki oyuncu arasindaki izlenen
yolu f degiskenine gore bulur.
                                                                                    //f 1
iken aranacak yoldaki aralık maximum 6 olur.0 oldugunda ise 100 olur.
            int fIndex=search(g,first);//ilk verilen ismin grafta aranmasi
            int sIndex=search(g,second);//ikinci verilen ismin grafta aranmasi

            if(g->artist[fIndex] == NULL || (g->artist[fIndex] != NULL && strcmp(g->artist[fIndex]->data,first)
!= 0)){
                        printf("Tabloda aranan ilk kisi bulunamadi.\n");//ilk verilen ismin grafta yoksa
                        return;
```

```
	}

	if(g->artist[sIndex] == NULL || (g->artist[sIndex] != NULL && strcmp(g->artist[sIndex]->data,second) != 0)){

		printf("Tabloda aranan ikinci kisi bulunamadi.\n");//ikinci verilen ismin grafta yoksa

		return;

	}

	//iki oyuncuda grafta varsa

	int count;//aranan iki isim icin maximum derinligi tutar.

	int flag=1;//eger aranan ikinci kisi bulunduysa aramayi kesmek icin kullanilmistir.

	int i;//iterasyon amacli kullanilmistir.

	char visited[TABLE_SIZE];//gezilen node larin bilgisini tutar.

	struct queue *q=(struct queue *)malloc(sizeof(struct queue));//grafta node lar bfs ile gezilmesi icin kullanilmistir.

	struct stack *s=(struct stack*)malloc(sizeof(struct stack));//grafta node lar bfs ile gezilirken sonradan eklenen diger oyunculari tutmak icin kullanilmistir.

	struct stack *cur;//queue dan cikan degeri uzerinde gezme yapilmasi icin kullanilmistir.

	struct node *iter;//queue dan cikan stack in en ustundeki kisinin oynadigi filmleri gezmek icin kullanilmistir.

	struct node *iter2;//queue dan cikan stack in en ustundeki kisinin oynadigi filmlerde oynayan oyunculari stack e eklemek icin kullanilmistir.



	q->front=NULL;//queue yu ilk basta bostur.

	q->rear=NULL;

	s->next=NULL;//stack ilk basta bostur.

	s->top=NULL;

	s->len=0;



	if(f == 1){

		count=6;

	}

	else{

		count=100;
```

```
	}
	for(i=0;i<TABLE_SIZE;i++){//hicbir node gezilmedi.
		visited[i]=0;
	}
	visited[fIndex]=1;//aranan node gezildi olarak isaretlendi
	push(s,first,fIndex);//node stack e atildi
	enqueue(q,s);//bu stack queue ya atildi.
	pop(s);//stack sifirlandi.
	while(q->front != NULL && flag){
		cur=dequeue(q);//queue nun basindaki eleman queue dan cekildi.
		if(cur->len / 2 > count){//maximum aranan derinlikte bulunamadi ise cikar.
			flag=0;
			printf("Aranan uzaklikta bulunamadi.\n");
		}
		else{
			if(strcmp(cur->top->data,second) == 0){//aranan kisi bulunduysa stack yazdirilir ve program biter.
				printf("Yolun uzunlugu: %d ve bulunan yol:\n",cur->len/2);
				print_stack(cur);
				flag=0;
			}
			else{
				iter=g->artist[cur->top->index]->next;//kuyrugun basindaki adamin oynadıgi filmler
				while(iter != NULL){
					iter2=g->film[iter->index]->next;//o filmde oynayan oyuncular
					while(iter2 != NULL){
						if(visited[iter2->index] == 0){
							push(cur,iter->data,iter->index);//film stack atilir
							push(cur,iter2->data,iter2->index);//oyuncu stack e atilir.
```

```
                                                enqueue(q,cur);//degisim yapilan stack queue ya atilir.
                                                visited[iter2->index]=1;//gezilen oyuncu gezildi olarak isaretlenir.
                                                pop(cur);//stacke eklenen oyuncu cikarilir.
                                                pop(cur);//stacke eklenen film cikarilir ve stack ilk haline doner.
                                        }
                                        iter2=iter2->next;
                                }
                                iter=iter->next;

                        }
                }
        }
}
int search(Graph *g,char *str){//graftaki oyuncular hashlenerek eklendigi icin eklenecegi indexi dondurur.
        int i=0;//bakilan yer dolu ise kacinci seferde bulundugu bilgisini tutar.
        int flag=1;//bakilan yerde aranan str kelimesi varsa aramayi kesmesi icin kullanilmistir.
        int key=get_key(str);//verilen str kelimesine ait key degerini dondurur.
        int index=funcH(key,i++);//bu keyin geldigi index bilgisini tutar.
        while(g->artist[index] != NULL && flag){
                if(strcmp(g->artist[index]->data,str) == 0)//eger aranan yerde str kelimesi varsa flag 0 yapilir ve arama biter.
                        flag=0;
                else{
                        index=funcH(key,i++);//aranan str kelimesi bulunamadiysa veya bos yer bulunamadiysa aramaya devam eder.
                }
        }
        return index;//bulunan indexi dondurur.
```

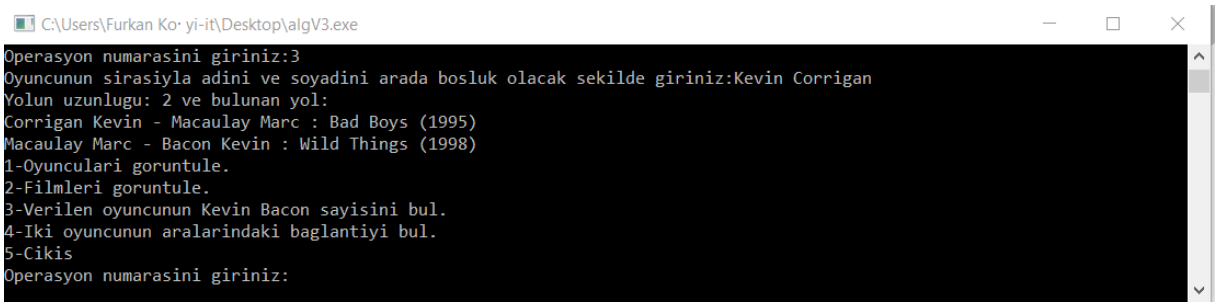
```
}
int get_key(char *word){//verilen kelimeye ait key degerini dondurur.
	int i;//iterasyon amacli kullanilmistir.
	int n=strlen(word);//verilen word adli kelimenin uzunluk bilgisini tasir.
	int key=0;//key verilen kelimenin ascii degerleri toplanarak bulunur.
	for(i=0;i<n;i++){
		key+=word[i];
	}
	return key;//key degeri dondurulur.
}
int funcH(int key,int i){//verilen key in tabloda yerlesecegi yeri dondurur.
	int x=key*key;
	return (x + i* i) % TABLE_SIZE;
}
void readFileFillGraph(char *filename,Graph *g){//verilen grafi verilen dosyayi okuyarak doldurur.
	char name[128];//okunan kelimeyi gecici olarak saklamak icin kullanilmistir.
	char ch;//dosyanin karakter karakter okunmasi icin kullanilmistir.
	int i=0;//name karakter dizisinde iterasyon amacli kullanilmistir.
	int j=0;//kac tane film eklendigi bigisini tutar.
	int flag=1;//okunan kelimenin film olup olmadiginin bilgisini tasir.
	struct node *newNode;//okunan kelimenin grafa eklenmesi icin kullanilmistir.
	FILE *fp=fopen(filename,"r");//verilen dosya ismini acmak icin kullanilmistir.
	if(fp == NULL){
		printf("Dosya acilamadi.\n");
		return;
	}
	while((ch=fgetc(fp)) != EOF){//dosyanin sonuna gelinene kadar oku
		if(ch != ',' && ch != '\n' && ch != '/'){
			name[i++]=ch;
			name[i]='\0';
```

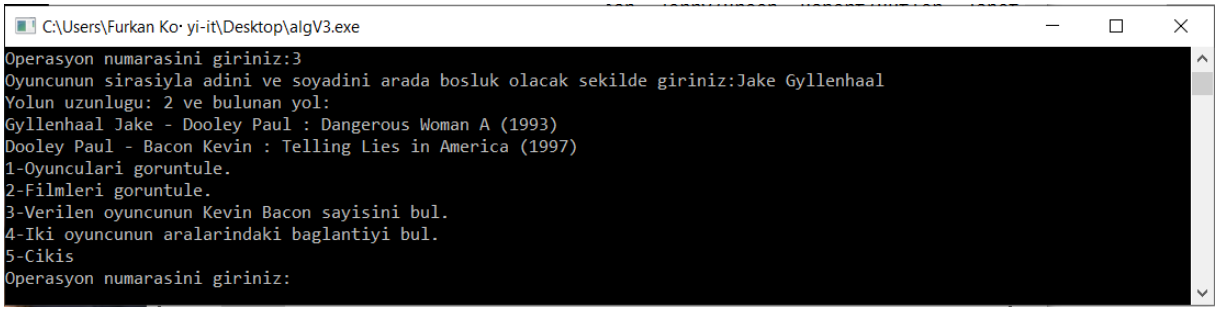
```
                }
                if(ch == '\n'){
                        addEdge(g,newNode->index,name);
                        i=0;
                        flag=1;
                }
                if(ch == '/'){
                        i=0;
                        if(flag){//okunan kelime film ise grafa direkt ekle
                                flag=0;
                                newNode=createNode(name);
                                newNode->index=j;
                                g->film[newNode->index]=newNode;
                                j++;//film sayisini arttir.
                        }
                        else{//okunan kelime oyuncu ise
                                addEdge(g,newNode->index,name);//grafa ekle ve aralarinda baglanti
olustur.
                        }
                }
        }
        fclose(fp);//dosyayi kapat.
}
int main(){
        Graph *g=createGraph();//graf olustur.
        char filename[32];//dosya adini saklamak icin kullanilmistir.
        char name[32];//ilk oyuncunun adini saklamak icin kullanilmistir.
        char name2[32];//ikinci oyuncunun adini saklamak icin kullanilmistir.
        char *token;//oyuncunun adini ve soyadini ayirmak icin kullanilmistir.
        char *token2;//oyuncunun adini ve soyadini ayirmak icin kullanilmistir.
        char *lname;//ilk oyuncunun adi ve soyadini yerd degistirmek icin kullanilmistir.
```

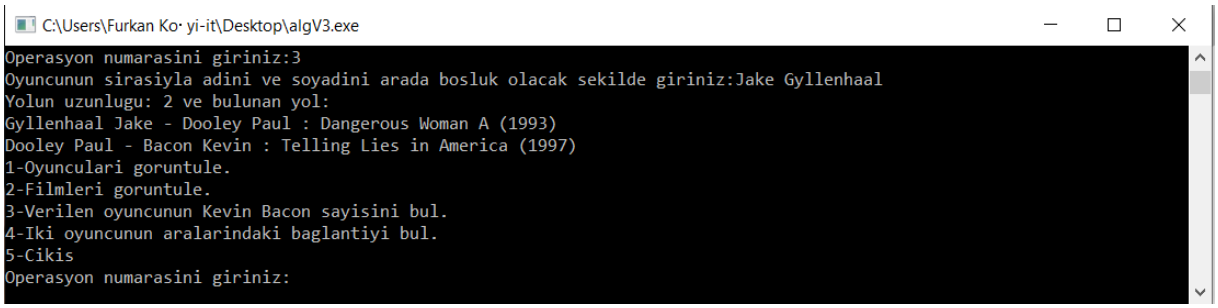
```
	char *lname2;//ikinci oyuncunun adi ve soyadini yerd degistirmek icin kullanilmistir.
	printf("Dosyanin ismini giriniz :");
	scanf("%s",filename);
	printf("Veritabani yukleniyor.\n");
	readFileFillGraph(filename,g);
	printf("Veritabani yuklendi.\n");
	int x=1;//while dongusunden cikmak icin kullanilmistir
	int option;//hangi operasyonun secildigini okumak icin kullanilmistir
	while(x){
		printf("1-Oyunculari goruntule.\n");
		printf("2-Filmleri goruntule.\n");
		printf("3-Verilen oyuncunun Kevin Bacon sayisini bul.\n");
		printf("4-Iki oyuncunun aralarindaki baglantiyi bul.\n");
		printf("5-Cikis\n");
		printf("Operasyon numarasini giriniz:");
		scanf("%d",&option);
		if(option == 1){
			printArtists(g);
		}
		else if(option == 2){
			printFilms(g);
		}
		else if(option == 3){
			printf("Oyuncunun sirasiyla soyadini ve adini arada bosluk olacak sekilde
giriniz:");
			getchar();
			gets(name);
			token=strtok(name," ");
			token2=strtok(NULL," ");
			lname=strcat(token2," ");
			lname=strcat(lname,token);
```

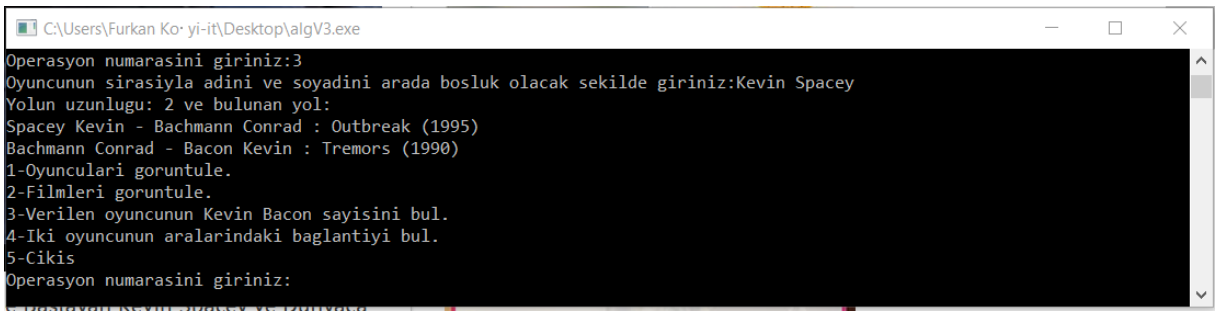
```
                bfs(g,"Bacon Kevin",lname,1);
            }
            else if(option == 4){
                printf("Ilk oyuncunun sirasiyla soyadini ve adini arada bosluk olacak sekilde
giriniz:");
                getchar();
                gets(name);
                token=strtok(name," ");
                token2=strtok(NULL," ");
                lname=strcat(token2," ");
                lname=strcat(lname,token);
                printf("Ikinci oyuncunun sirasiyla soyadini ve adini arada bosluk olacak sekilde
giriniz:");
                gets(name2);
                token=strtok(name2," ");
                token2=strtok(NULL," ");
                lname2=strcat(token2," ");
                lname2=strcat(lname2,token);
                bfs(g,lname,lname2,0);
            }
            else if(option == 5){
                x=0;
            }
            else{
                printf("Lutfen Gecerli bir operasyon giriniz.\n");
                printf("\n");
            }
        }
        return 0;
}
```

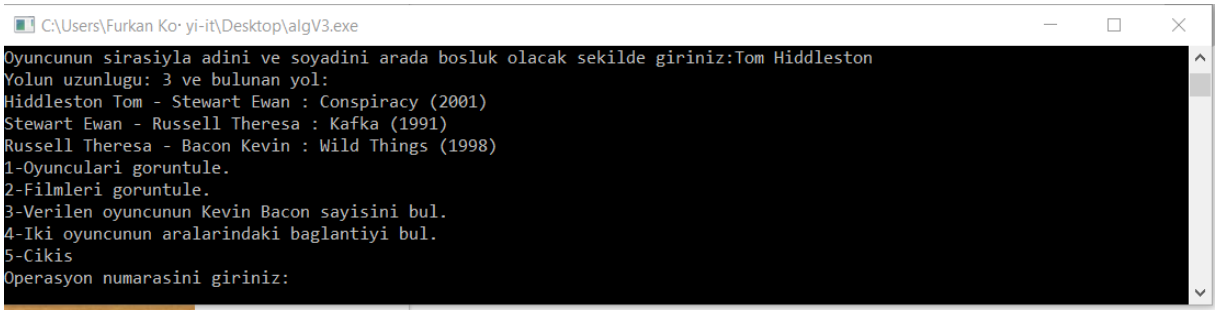
Ekran Çıktıları:

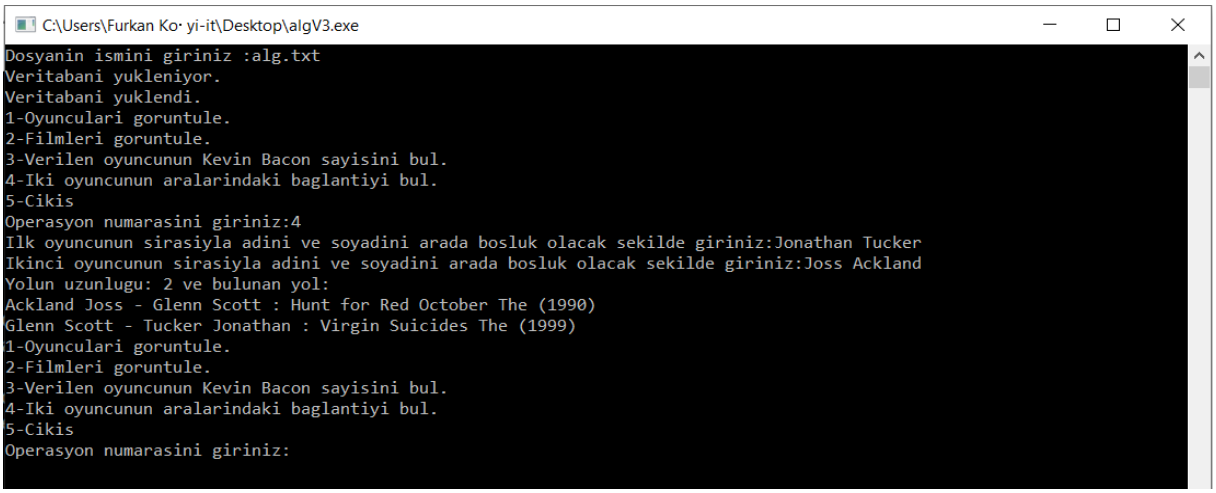


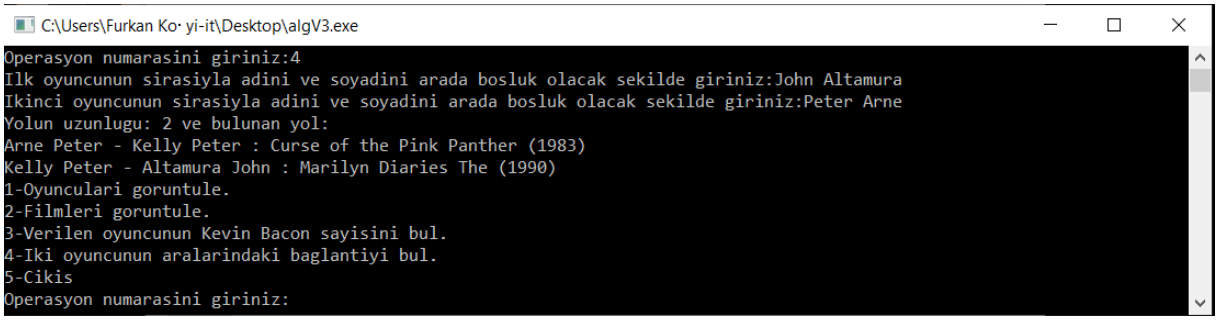


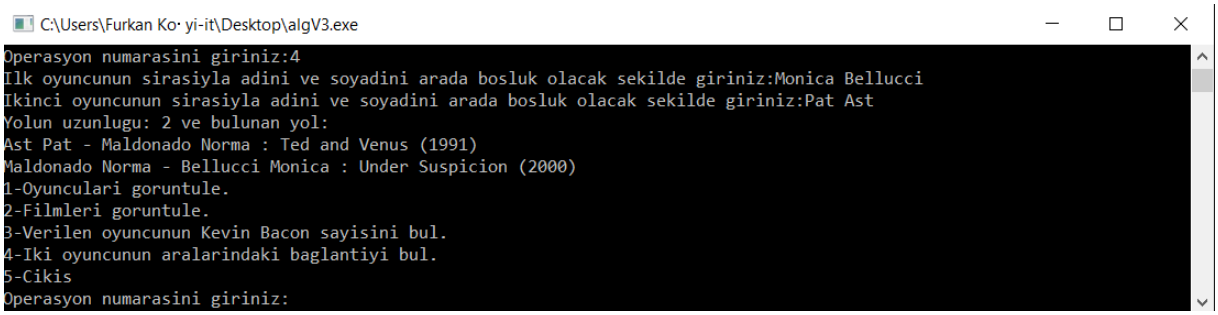


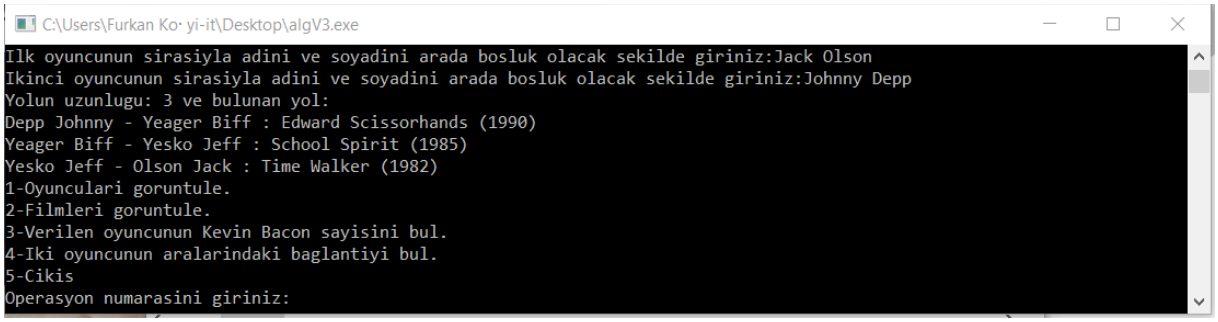


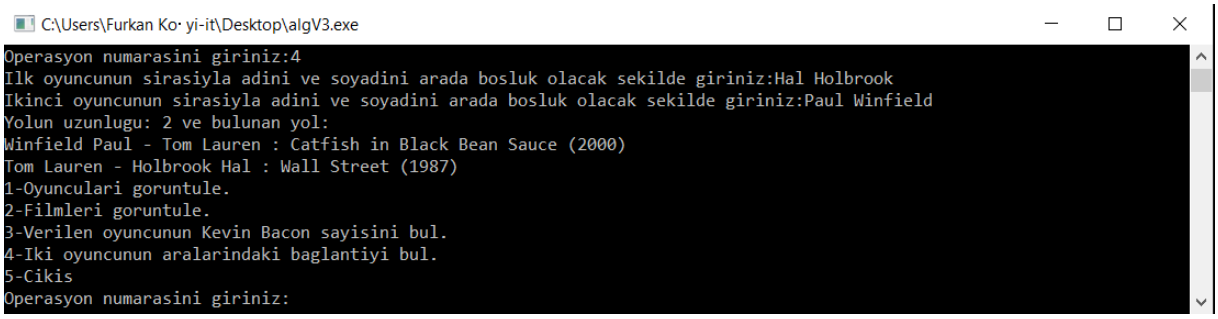



C:\Users\Furkan Ko· yi-it\Desktop\algV3.exe
Dosyanin ismini giriniz :alg.txt
Veritabani yukleniyor.
Veritabani yuklendi.
1-Oyunculari goruntule.
2-Filmleri goruntule.
3-Verilen oyuncunun Kevin Bacon sayisini bul.
4-Iki oyuncunun aralarindaki baglantiyi bul.
5-Cikis
Operasyon numarasini giriniz:4
Ilk oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Jonathan Tucker
Ikinci oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Joss Ackland
Yolun uzunlugu: 2 ve bulunan yol:
Ackland Joss - Glenn Scott : Hunt for Red October The (1990)
Glenn Scott - Tucker Jonathan : Virgin Suicides The (1999)
1-Oyunculari goruntule.
2-Filmleri goruntule.
3-Verilen oyuncunun Kevin Bacon sayisini bul.
4-Iki oyuncunun aralarindaki baglantiyi bul.
5-Cikis
Operasyon numarasini giriniz:


C:\Users\Furkan Ko· yi-it\Desktop\algV3.exe
Operasyon numarasini giriniz:4
Ilk oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:John Altamura
Ikinci oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Peter Arne
Yolun uzunlugu: 2 ve bulunan yol:
Arne Peter - Kelly Peter : Curse of the Pink Panther (1983)
Kelly Peter - Altamura John : Marilyn Diaries The (1990)
1-Oyunculari goruntule.
2-Filmleri goruntule.
3-Verilen oyuncunun Kevin Bacon sayisini bul.
4-Iki oyuncunun aralarindaki baglantiyi bul.
5-Cikis
Operasyon numarasini giriniz:


C:\Users\Furkan Ko· yi-it\Desktop\algV3.exe
Operasyon numarasini giriniz:4
Ilk oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Monica Bellucci
Ikinci oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Pat Ast
Yolun uzunlugu: 2 ve bulunan yol:
Ast Pat - Maldonado Norma : Ted and Venus (1991)
Maldonado Norma - Bellucci Monica : Under Suspicion (2000)
1-Oyunculari goruntule.
2-Filmleri goruntule.
3-Verilen oyuncunun Kevin Bacon sayisini bul.
4-Iki oyuncunun aralarindaki baglantiyi bul.
5-Cikis
Operasyon numarasini giriniz:


C:\Users\Furkan Ko· yi-it\Desktop\algV3.exe
Ilk oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Jack Olson
Ikinci oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Johnny Depp
Yolun uzunlugu: 3 ve bulunan yol:
Depp Johnny - Yeager Biff : Edward Scissorhands (1990)
Yeager Biff - Yesko Jeff : School Spirit (1985)
Yesko Jeff - Olson Jack : Time Walker (1982)
1-Oyunculari goruntule.
2-Filmleri goruntule.
3-Verilen oyuncunun Kevin Bacon sayisini bul.
4-Iki oyuncunun aralarindaki baglantiyi bul.
5-Cikis
Operasyon numarasini giriniz:


C:\Users\Furkan Ko· yi-it\Desktop\algV3.exe
Operasyon numarasini giriniz:4
Ilk oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Hal Holbrook
Ikinci oyuncunun sirasiyla adini ve soyadini arada bosluk olacak sekilde giriniz:Paul Winfield
Yolun uzunlugu: 2 ve bulunan yol:
Winfield Paul - Tom Lauren : Catfish in Black Bean Sauce (2000)
Tom Lauren - Holbrook Hal : Wall Street (1987)
1-Oyunculari goruntule.
2-Filmleri goruntule.
3-Verilen oyuncunun Kevin Bacon sayisini bul.
4-Iki oyuncunun aralarindaki baglantiyi bul.
5-Cikis
Operasyon numarasini giriniz: