Kaba Kod

# Projede Kullanılan Tüm Fonksiyonlar

### Menu Fonksiyonları

1. Menu() → Menu Ekrani
2. sehirAra() -> Sehir Arama menusu
3. komsuIslemMenu(struct node \*iter,sehirPlaka) -> Komsu islemleri menusu
4. sehirSil() -> Sehir silme menusu
5. komsuSil(struct node \*r) -> Komsu Silme menusu
6. sehirDuzenle() -> Sehir duzenleme menusu

### **Arama** Fonksiyonları

1. isimArama() -> Isime gore sehir arama
2. plakaArama() -> Plakaya gore sehir arama
3. bolgeAra() -> Bolge kisaltmasina gore sehirleri arama
4. kriterArama() -> Kriterlere gore sehir arama
5. komsuPlakaBul(struct node \*r, char sehirAdi[20]) -> Komsunun plakasinin bulunmasi
6. sehirToPlaka(char sehirAdi[20]) -> Sehir Adini Plaka numarasina cevirme

### **Okuma** Fonksiyonları

1. dosyaOku(status) -> dosyanin Okunma islemi
2. parcala(char kelime[200],status) -> Dosyadan gelen satirin parcalanma islemi
3. dosyaYazma() -> Outputs yazma islemi

### **Ekleme** Fonksiyonları

1. struct node \* ekle(struct node \* r,plaka,char sehirAdi[20],char bolge[5],komsu\_sayisi) -> Sehir ekleme
2. struct komsuNode \* komekle(struct komsuNode\* r,komPlaka,sehirPlaka) -> Komsu Ekleme
3. sehirEkle() -> Sehir ekleme menusu

### **Ekrana Çıktı Verme** Fonksiyonları

1. ekranaYaz(struct node\* r) -> Ekrana bilgileri cikartir
2. renkDegistir(renk) -> Ekran rengini degistirir
3. menuAdiOlustur(char menuAdi[100],renk,son) -> Menu olusturur
4. sehirGetir(plaka) -> Sehir bilgisini getirir
5. komsulariniGetir(struct node \* r) -> Komsularini ekrana getirir

### **Silme** Fonksiyonları

1. komsuSilAuto(struct node \*r,plaka) ->Otomatik komsu silme
2. struct node \* sil(struct node \* r,struct node \*iter,plaka) -> Sehir silme
3. struct komsuNode \* komsuyusil(struct komsuNode \* kroot,struct komsuNode \* kiter,struct node \* r,plaka) ->komsu silme

### Kontrol Fonksiyonları

1. komsuKontrol(struct node\* r,plaka,sehirPlaka) -> Komsu kontrol
2. sehirIsimKontrol(struct node \*r,char sehirAdi[20]) -> Ayni isme ait sehir var mi
3. komsuCek() -> Komsularini ceker
4. bolgeRenkleri(char bolge[5]) -> Bolge renklendirir

# Menu Fonksiyonlarının Kaba Kodları

## Menu:

* int secim;
* Menu yazilarini ekrana yaz
* ekrandan alınan değeri seçim değişkenine aktar
* secim = 1 ise ekranaYaz(root) fonksiyonuna git.
* secim = 2 ise sehirEkle fonksiyonuna git.
* secim = 3 ise sehirSil() fonksiyonuna git.
* secim = 4 ise sehirDuzenle() fonksiyonuna git.
* secim = 5 ise sehirAra() fonksiyonuna git.
* secim = 9 ise projeyi kapat

## SehirAra

* int secim;
* Menu yazilarini ekrana yaz
* ekrandan alınan değeri seçim değişkenine aktar
* secim = 1 ise isimArama() fonksiyonuna git.
* secim = 2 ise plakaArama() fonksiyonuna git.
* secim = 3 ise bolgeAra() fonksiyonuna git.
* secim = 4 ise kriterArama() fonksiyonuna git.

## komsuIslemMenu(struct node \*iter,sehirPlaka)

* int secim;
* Menu yazilarini ekrana yaz
* ekrandan alınan değeri seçim değişkenine aktar
* secim = 1 ise komsuSil(iter) fonksiyonuna git.
* secim = 2 ise komsuEkle(iter,sehirPlaka) fonksiyonuna git.

## SehirSil

* int secim;
* Menu yazilarini ekrana yaz
* ekrandan alınan değeri seçim değişkenine aktar
* secim = 1
  + Kullanıcıdan plaka degerini al
  + node ekranında plakayı kontrol et
  + Plaka bulunursa
    - Şehrin bilgilerini ekrana yaz
    - Kullanıcıya sor. Plakayı silmek istiyor musunuz ?
    - HAYIR
      * Döngüden çık
    - EVET
      * Plakaya ait tüm komşu şehirleri bul ve sil.
      * Root = sil(root,iter,plaka) degerlerini gönder.
      * Şehir Silindi.
* Secim = 2
  + Kullanıcıdan şehrin ismini al
  + node ekranında sehir ismini arayarak plakayı kontrol et
  + Plaka bulunursa
    - Şehrin bilgilerini ekrana yaz
    - Kullanıcıya sor. Şehri silmek istiyor musunuz ?
    - HAYIR
      * Döngüden çık
    - EVET
      * Plakaya ait tüm komşu şehirleri bul ve sil.
      * Root = sil(root,iter,plaka) degerlerini gönder.
      * Şehir Silindi.
  + Eğer şehirler bulunamazsa
    - Şehir bulunamadı hata mesajını ekrana yaz.

## KomsuSil

* int secim;
* int plaka = -999;
* char sehirAdi[20];
* komsuSilGo:
* Menu yazilarini ekrana yaz
* ekrandan alınan değeri seçim değişkenine aktar
* secim = 1 ise
  + Şehrin ismini kullanıcıdan al
  + Sistemden kullanıcıdan alınan şehrin plakasını bul
  + plaka degiskenine aktar.
* secim = 2 ise
  + Şehrin plakasını kullanıcıdan al
* secim = 0 ise
  + menu() fonksiyonuna geri git.
* struct komsuNode \* kiter;
* struct komsuNode \* kroot;
* kroot = (void \*)r->komsuNode;
* kiter = (void \*)r→komsuNode;
* int status;
* Komsunun node’daki yerini bul
  + Komşu bulunursa status degerini 1 yap
* Status = 1 ise
  + kroot = komsuyusil(kroot,kiter,r,plaka) Fonksiyonu ile komşuyu sil
  + Silinen komşusun plakasını al ve o şehire git
  + komsuSilAuto(iter,r→plaka) Fonksiyonu ile o şehrin komşusundan da ana şehri sil
  + Kalan tüm komşuları ekrana yaz.
* Status = 0 ise
  + Ekrana: Komşu bulunamadı yaz.

## SehirDuzenle

* int secim;
* int plaka = 0;
* char sehirAdi[20];
* int secim = 0;
* int sonuc = 0;
* Menu yazilarini ekrana yaz
* ekrandan alınan değeri seçim değişkenine aktar
* secim = 1 ise
  + Kullanıcıdan plaka değerini al.
  + Plakanın sistemde olup olmadığını kontrol et:
    - sonuc = sehirKontrol(root,plaka);
* secim = 2 ise
  + Kullanıcıdan şehir adını al.
  + Şehrin sistemde olup olmadığını kontrol et:
    - sonuc = sehirIsimKontrol(root,sehirAdi);
* sonuc = 1
  + secim = 2
    - şehrin adından şehrin plakasını bul.
    - Plaka değerine bu plakayı aktar.
* char sehirAdi[20];
* char bolge[5];
* int yeniPlaka;
* Kullanıcıdan gerekli değerleri al ve değişkenlere aktar.
* Girilen yeni plaka kodunu al ve kontrol et
  + yeniPlakaKontrol = sehirKontrol(root,yeniPlaka);
* Girilen yeni şehir adını al ve kontrol et
  + yeniIsimKontrol = sehirIsimKontrol(root,sehirAdi);
* yeniPlakaKontrol = 1 ise
  + Yeni plakaya ait baska bir sehir var.
  + sehirDuzenle() fonksiyonuna geri git.
* YeniIsimKontrol == 1 ise
  + Yeni isime ait baska bir sehir var
  + sehirDuzenle() fonksiyonuna geri git.
* yeniPlakaKontrol = 0 ve yeniIsimKontrol = 0 ise
  + Girilen veriler ile yeni bir şehir ekle
    - root = ekle(root,yeniPlaka,sehirAdi,bolge,0);
  + struct node \*yeniSehir;
  + struct node \*eskiSehir;
  + yeniSehir = root;
  + eskiSehir = root;
  + yeni eklenen sehrin plakasını arayarak o nodu bul.
  + Değiştirilecek olan şehrin plakasını arayarak o nodu bul.
  + Yeni sehirin komsu nodu na eski sehrin komsu nodunu ekle
  + yeni sehrin komsu sayısına eski sehrin komsu sayısını ekle
  + eski sehri sil
    - sil(root,eskiSehir,plaka);
    - eski sehrin komsu olarak kayıtlı olduğu sehirlerden komşuluğu sil ve yeni şehrin plakasını ekle
* sonuc = 0 ise (Yani düzenlenmesini istediğiniz şehir sistemde yok ise)
  + Sehir yok eklemek ister misiniz? Evet ise 1 Hayır ise 0
    - soru = 0 ise
      * döngüden çık
    - soru = 1 ise
      * sehirEkle() fonksiyonuna git.

# Arama Fonksiyonlarının Kaba Kodları

## IsimArama

* char sehirAdi[20];
* Kullanıcıdan şehir Adını al.
* struct node \* iter;
* iter = root;
* int searchStatus = 0;
* Node değişkeni null olana kadar devam et
  + şehir Adı girilen şehir adına eşit ise
    - bolgeRenkleri(iter→bolge) fonksiyonuna giderek rengini belirle.
    - Şehrin bilgilerini ekrana yaz
    - Komşularını getir ve ekrana yaz.
    - SearchStatus = 1
    - komsuIslemMenu(iter,iter→plaka) fonksiyonunu çağır.
* SearchStatus = 0 ise
  + Sehir bulunamadi. Eklemek Istermisiniz ? 0 Hayir, 1 Evet:
  + int status;
  + status = 0 ise
    - döngüden çık
  + status = 1 ise
    - sehirEkle() fonksiyonuna git.

## plakaArama

* Int plaka;
* Kullanıcıdan plaka al.
* struct node \* iter;
* iter = root;
* int searchStatus = 0;
* Node değişkeni null olana kadar devam et
  + plaka girilen plaka eşit ise
    - bolgeRenkleri(iter→bolge) fonksiyonuna giderek rengini belirle.
    - Şehrin bilgilerini ekrana yaz
    - Komşularını getir ve ekrana yaz.
    - SearchStatus = 1
    - komsuIslemMenu(iter,iter→plaka) fonksiyonunu çağır.
* SearchStatus = 0 ise
  + Sehir bulunamadi. Eklemek Istermisiniz ? 0 Hayir, 1 Evet:
  + int status;
  + status = 0 ise
    - döngüden çık
  + status = 1 ise

sehirEkle() fonksiyonuna git.

## BolgeAra

* char bolge[5];
* Kullanıcıdan bölge kısaltmasını al.
* int searchStatus = 0;
* struct node \* iter;
* iter = root;
* iter != null olana kadar döngüye devam et
  + kullanıcıdan alınan bölge ile iter→bölge eşit ise şehir bilgilerini ekrana yaz
  + bolgeRenkleri(iter->bolge);
  + searchStatıs = 1 yap.
* SearchStatus = 0 ise
  + Bolge bulunamadi

## kriterArama

* Int min,max;
* Kullanıcıdan min ve max değerlerini al.
* Max = 0 ise
  + max = 999999
* struct node \* iter;
* iter = root;
* struct node \* sehirSay;
* sehirSay = root;
* int sehirSayisi = 0;
* sehirSay = NULL olana kadar
  + sehirSayisi++
* int plakalar[sehirSayisi][max];
* plakalar matrisinin tüm elemanlarına -99 degerini ver
* int kosulluSehirler = 0;
* iter = NULL olana kadar
  + iter->komsu\_sayisi >= min && iter->komsu\_sayisi<= max
    - bolgeRenkleri(iter→bolge)
    - Sehir bilgilerini ekrana yaz
    - struct komsuNode \* komNode;
    - komNode = (void \*)iter→komsuNode;
    - plakalar[kosulluSehirler][0] = iter→plaka;
    - int komSehirler = 1;
    - komNode = NULL olana kadar
      * plakalar[kosulluSehirler][komSehirler] = komNode→komPlaka;
      * komSehirler++;
    - kosulluSehirler++;
* int ozelSecim;
* Ozel Arama Yapma Ister Misiniz ? (Hayir 0) (Evet 1)
* ozelSecim = 1 ise
  + int plakaOrIsim;
  + Menu Olustur
  + plakaOrIsim = 1 ise
    - int temp = 0;
    - temp != 1
      * char tempSehirAdi[20];
      * Sehir adini giriniz: Tamamlandiysa 0 giriniz:
      * tempSehirAdi[0] = 0 ise
        + temp = 1
      * tempSehir[0] != 0 ise
        + int tempPlaka = sehirToPlaka(tempSehirAdi);
        + tempPlaka == -99 ise

sehir sistemde kayitli degildir.

Goto hataliSecim;

* + - * + KriterPlakalar[sayac] = tempPlaka;
        + sayac++;
* ozelSecim = 2 ise
  + int temp = 0;
  + while(temp != 1)
    - int tempPlaka = -999;
    - Sehir plakasini giriniz: Tamamlandiysa 0 giriniz:
    - tempPlaka = 0 ise
      * temp = 1;
    - tempPlaka != 0 ise
      * sehirKontrol(root,tempPlaka) = 0 ise
        + plakali sehir sistemde kayitli degildir.
        + goto hataliSecim;
      * KriterPlakalar[sayac] = tempPlaka;
      * sayac++;
    - int i = 0;
    - while(i < sayac)
      * if(sehirKontrol(root,KriterPlakalar[i]) == 0)
        + Boyle bir sehir sistemde yok
        + Arama menusune yonlendiriliyorsunuz
        + sehirAra();
    - i++;
* int yokMesaji = 1;
* KriterPlakalar[0] != -99 ise
  + Aranan Sehirler
* for(int i = 0; i<sayac; i++)
  + sehirGetir(KriterPlakalar[i]);
* Ortak Sehirler Listesi
  + for(int i = 0; i< sehirSayisi; i++)
  + int varMi = 0;
  + for(int k = 1; k< max; k++)
    - for(int l = 0; l< sayac; l++)
      * for(int l = 0; l< sayac; l++)
        + if(plakalar[i][k] == KriterPlakalar[l])

varMi++

* + - * if(varMi == sayac)
        + sehirGetir(plakalar[i][0]);
        + yokMesaji = 0;
      * varMi = 0;
  + yokMesaji = 1 ve KriterPlakalar[0] = -99 ise
    - Bu kriterlere uygun ortak sehirler bulunamadi

## KomsuPlakaBul(struct node \*r, char sehirAdi[20])

* struct node \* iter;
* iter = r
* while(iter != NULL)
  + iter→sehirAdi = sehirAdi ise
    - return iter→plaka
* !!Dosya hatali!!
* exit(404);
* return -999

## sehirToPlaka(char sehirAdi[20])

* struct node \*iter;
* iter = root;
* while(iter != NULL)
  + iter→sehirAdi == sehirAdi ise
    - return iter→plaka;
* return -99

# Okuma Fonksiyonlarının Kaba Kodları

## dosyaOku(status)

* FILE \*dosya;
* dosya = fopen("sehirler.txt","r");
* if(dosya)
  + char birkelime[200];
  + int c;
  + do
    - c = fscanf(dosya,"%s",birkelime);
    - if (feof(dosya))
      * break;
    - if(status == 0)
      * parcala(birkelime,0);
    - else if(status == 1)
      * parcala(birkelime,1);
  + while (c != EOF);
  + fclose(dosya);
  + if(!dosya)
    - status == 0
    - Ana dizinde sehirler.txt dosyasi bulunamadi.!!!

## parcala(char kelime[200],status)

* const char s[4] = ",";
* char \*token;
* int plaka;
* char \*sehirAdi[20];
* char \*bolge[5];
* token = strtok(kelime, s);
* int a = 0;
* while( token != NULL )
  + if(status == 0)
    - if(a == 0)
      * plaka = atoi(token);
      * if(sehirKontrol(root,plaka) == 1)
        + Listede ayni sehirden birden fazla var.
        + plaka = 0;
    - if(a == 1)
      * \*sehirAdi = token;
    - if(a == 2)
      * \*bolge = token;
      * if(plaka != 0 && \*bolge != NULL && sehirAdi != NULL)
        + root = ekle(root,plaka,\*sehirAdi,\*bolge,0);
  + else if(status == 1)
  + int sehirPlaka;
  + if(a == 0)
    - sehirPlaka = atoi(token);
  + if(a > 2)
    - if(komsuKontrol(root,komsuPlakaBul(root,token),sehirPlaka) == 1)
      * Listede ayni sehirden birden fazla var.
    - else
      * komsuRoot = komekle(komsuRoot,komsuPlakaBul(root,token),sehirPlaka);
  + a++
  + token = strtok(NULL, s);

## DosyayaYazma

* time\_t t = time(NULL);
* struct tm \*tm = localtime(&t);
* system("mkdir outputs");
* yazilacakDosya = fopen("outputs/output.txt", "a");
* Dosya,"Olusturulma Tarihi: asctime™

# **Ekleme** Fonksiyonlarının Kaba Kodları

## struct node \* ekle(struct node \* r,plaka,char sehirAdi[20],char bolge[5],komsu\_sayisi)

* if(r== NULL)
  + r = (struct node\*)malloc(sizeof(struct node));
  + r->ileri = NULL;
  + r->geri = NULL;
  + r->plaka = plaka;
  + r->komsu\_sayisi = komsu\_sayisi;
  + strcpy(r→sehirAdi,sehirAdi);
  + strcpy(r→bolge,bolge);
  + r->komsuNode = NULL;
  + return r;
* else
  + if(r->plaka > plaka)
    - struct node \* temp = (struct node\*)malloc(sizeof(struct node));
    - temp->ileri = r;
    - temp->geri = NULL;
    - r->geri = temp;
    - temp->plaka = plaka;
    - temp->komsu\_sayisi = komsu\_sayisi;
    - strcpy(temp→sehirAdi,sehirAdi);
    - strcpy(temp→bolge,bolge);
    - temp->komsuNode = NULL;
    - return temp;
  + Else
    - struct node \* iter;
    - iter = r;
    - if(iter->ileri == NULL)
      * struct node \* temp = (struct node\*)malloc(sizeof(struct node));
      * iter->ileri = temp;
      * temp->geri = iter;
      * temp->ileri = NULL;
      * temp->plaka = plaka;
      * return r;
    - if(iter->ileri == NULL)
      * struct node \* temp = (struct node\*)malloc(sizeof(struct node));
      * iter->ileri = temp;
      * temp->geri = iter;
      * temp->ileri = NULL;
      * temp->plaka = plaka;
      * temp->komsu\_sayisi = komsu\_sayisi;
      * strcpy(temp->sehirAdi,sehirAdi);
      * strcpy(temp->bolge,bolge);
      * temp->komsuNode = NULL;
    - else
      * struct node \* temp = (struct node\*)malloc(sizeof(struct node));
      * iter->ileri->geri = temp;
      * temp->ileri = iter->ileri;
      * iter->ileri = temp;
      * temp->geri = iter;
      * temp->plaka = plaka;
      * temp->komsu\_sayisi = komsu\_sayisi;
      * strcpy(temp->sehirAdi,sehirAdi);
      * strcpy(temp->bolge,bolge);
      * temp->komsuNode = NULL;
      * return r;
* return r;

## struct komsuNode \* komekle(struct komsuNode\* r,komPlaka,sehirPlaka)

* struct node \* Globaliter;
* Globaliter = root;
* while(Globaliter->plaka != sehirPlaka)
  + Globaliter = Globaliter→ileri;
* if(Globaliter->komsuNode == NULL)
  + r = (struct komsuNode\*)malloc(sizeof(struct komsuNode));
  + r->komIleri = NULL;
  + r->komPlaka = komPlaka;
  + Globaliter->komsuNode = (void \*)r;
  + Globaliter->komsu\_sayisi = Globaliter→komsu\_sayisi+1;
  + return r;
* else
  + if(r->komPlaka > komPlaka)
    - struct komsuNode \* temp = (struct komsuNode\*)malloc(sizeof(struct komsuNode));
    - temp->komIleri = r;
    - temp->komPlaka = komPlaka;
    - Globaliter->komsu\_sayisi = Globaliter→komsu\_sayisi+1;
    - Globaliter->komsuNode = (void \*)temp;
    - return temp;
  + else
    - struct komsuNode \* iter;
    - iter = r;
    - while(iter->komIleri !=NULL && iter->komIleri->komPlaka < komPlaka)
      * iter = iter→komIleri;
    - if(iter->komIleri == NULL)
      * struct komsuNode \* temp = (struct komsuNode\*)malloc(sizeof(struct komsuNode));
      * iter->komIleri = temp;
      * temp->komIleri = NULL;
      * temp->komPlaka = komPlaka;
      * Globaliter->komsu\_sayisi = Globaliter→komsu\_sayisi+1;
      * return r;
    - else
      * struct komsuNode \* temp = (struct komsuNode\*)malloc(sizeof(struct komsuNode));
      * temp->komIleri = iter->komIleri;
      * iter->komIleri = temp;
      * temp->komPlaka = komPlaka;
      * Globaliter->komsu\_sayisi = Globaliter→komsu\_sayisi+1;
      * return r;
* return r;

## sehirEkle

* int plaka;
* char sehirAdi[20];
* char bolge[5];
* Gerekli bilgileri kullanıcıdan al
* sehirKontrol(root,plaka) = 0 ise
  + sehirIsimKontrol(root,sehirAdi) = 0 ise
    - Şehri Ekle
    - root = ekle(root,plaka,sehirAdi,bolge,0);
    - sehirGetir(plaka);
  + else
    - Boyle bir isimde sehir var. Tekrar ekleyemessiniz
* sehirKontrol(root,plaka) = 1 ise
  + Boyle bir plakada sehir var. Tekrar ekleyemessiniz
* else
  + Plaka Hatali

# Ekrana Çıktı Verme Fonksiyonlarının Kaba Kodları

## ekranaYaz(struct node\* r)

* struct node \* iter;
* iter = r;
* iter = NULL ise
  + Liste Bos
* else
  + while(iter != NULL)
    - bolgeRenkleri(iter→bolge);
    - Şehir bilgilerini getir
    - struct komsuNode \*kom;
    - kom = (void \*)iter→komsuNode;
    - while(kom != NULL)
      * Komşu bilgilerini getir

## renkDegistir(renk)

* HANDLE hConsole;
* hConsole=GetStdHandle(STD\_OUTPUT\_HANDLE);
* SetConsoleTextAttribute(hConsole, renk);

## menuAdiOlustur(char menuAdi[100],renk,son)

NOT: Tasarımsal kodlar olmasından dolayı direk kodlar yazılmıştır.

* HANDLE hConsole;
* unsigned short int colors;
* hConsole=GetStdHandle(STD\_OUTPUT\_HANDLE);
* son = 0 ise
  + colors = 3;
  + SetConsoleTextAttribute(hConsole, colors);
  + printf("\t\t | ");
  + colors = renk;
  + SetConsoleTextAttribute(hConsole, colors);
  + printf("%s",menuAdi);
  + colors = 3;
  + SetConsoleTextAttribute(hConsole, colors);
  + printf(" |\n");
* son = 1 ise
  + colors = 3;
  + SetConsoleTextAttribute(hConsole, colors);
  + printf("\t\t ############################################################\n");
* son = 2 ise
  + colors = 3;
  + SetConsoleTextAttribute(hConsole, colors);
  + printf("\t\t ");
  + colors = renk;
  + SetConsoleTextAttribute(hConsole, colors);
  + printf("%s",menuAdi);
* SetConsoleTextAttribute(hConsole, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_RED);

## sehirGetir(plaka)

* struct node \*iter;
* iter = root;
* while(iter!= NULL)
  + iter->plaka = plaka ise
    - Şehrin bilgisini ekrana yaz

## komsulariniGetir(struct node \* r)

* struct komsuNode \* kNode;
* struct node \* iter;
* iter = (void \*)root;
* kNode = (void \*)r→komsuNode;
* while(kNode != NULL)
  + while(iter != NULL)
    - kNode->komPlaka = iter→plaka ise
      * Komşu bilgisini ekrana yaz

# **Silme** Fonksiyonlarının Kaba Kodları

## komsuSilAuto(struct node \*r,plaka)

* struct komsuNode \* kiter;
* struct komsuNode \* kroot;
* kroot = (void \*)r→komsuNode;
* kiter = (void \*)r→komsuNode;
* int status;
* while(kiter != NULL)
  + kiter->komPlaka = plaka ise
    - status =1;
* status = 1 ise
  + kroot = komsuyusil(kroot,kiter,r,plaka);

## struct node \* sil(struct node \* r,struct node \*iter,plaka)

* if(iter->ileri == NULL && iter->geri == NULL)
  + return NULL;
* if(iter == root)
  + struct node \*temp;
  + temp = iter→ileri;
  + iter->ileri = NULL;
  + temp->geri = NULL;
  + free(iter);
  + return temp;
* else
  + if(iter->ileri != NULL)
    - struct node \*tempIleri;
    - struct node \*tempGeri;
    - tempIleri = iter→ileri;
    - tempGeri = iter→geri;
    - iter->ileri = NULL;
    - iter->geri = NULL;
    - tempGeri->ileri = tempIleri;
    - tempIleri->geri = tempGeri;
    - free(iter);
  + else
    - struct node \*tempGeri;
    - tempGeri = iter→geri;
    - iter->geri = NULL;
    - tempGeri->ileri = NULL;
    - free(iter);
* return r;

## struct komsuNode \* komsuyusil(struct komsuNode \* kroot,struct komsuNode \* kiter,struct node \* r,plaka)

* if(kiter->komPlaka == kroot→komPlaka)
  + struct komsuNode \* temp;
  + temp = kiter→komIleri;
  + kiter->komIleri = NULL;
  + free(kiter);
  + r->komsuNode = (void \*)temp;
  + r->komsu\_sayisi = r->komsu\_sayisi – 1;
  + return temp;
* else
  + struct komsuNode \* niter;
  + niter = kroot;
  + while(niter->komIleri != NULL)
    - if(niter->komIleri->komPlaka == plaka)
      * döngüden çık
  + if(niter->komIleri != NULL)
    - struct komsuNode \* temp;
    - temp = kiter→komIleri;
    - niter->komIleri = temp;
    - kiter->komIleri = NULL;
    - free(kiter);
    - r->komsu\_sayisi = r->komsu\_sayisi – 1;
    - r->komsuNode = (void \*)kroot;
    - return kroot;
  + else if(niter->komIleri == NULL)
    - free(niter→komIleri);
    - r->komsuNode = (void \*)kroot;
    - r->komsu\_sayisi = r->komsu\_sayisi – 1;
    - return kroot;
* return kroot;

# **Kontrol** Fonksiyonlarının Kaba Kodları

## komsuKontrol(struct node\* r,plaka,sehirPlaka)

* struct node \* iter;
* iter = r;
* struct komsuNode \* komIter;
* int status = 0; //0 sehir yok 1 sehir var 2 plaka hatali
* while(iter->plaka != sehirPlaka)
  + iter = iter→ileri;
* komIter = (void \*)iter→komsuNode;
* if(komIter == NULL)
  + status = 0;
* while(komIter!=NULL)
  + if(komIter->komPlaka== plaka)
    - status = 1;
* return status;

## sehirIsimKontrol(struct node \*r,char sehirAdi[20])

* struct node \* iter;
* iter = r;
* int status = 0; //0 sehir yok 1 sehir var
* while(iter != NULL)
  + if(strcasecmp(iter->sehirAdi,sehirAdi) == 0)
    - status = 1;
    - döngüden çık
* return status;

## komsuCek()

* DosyaOku(1) fonksiyonuna 1 değerini gönder.

## bolgeRenkleri(char bolge[5])

* Bolge = MA ise → renkDegistir(1)
* Bolge = IA ise → renkDegistir(14)
* Bolge = KA ise → renkDegistir(2)
* Bolge = AK ise → renkDegistir(4)
* Bolge = DA ise → renkDegistir(5)
* Bolge = EG ise → renkDegistir(15)
* Bolge = GA ise → renkDegistir(11)
* else → renkDegistir(15);