

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2023–2024 FALL SEMESTER**

**PROJECT-2 REPORT**

**LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI**

**DELIVERY DATE**

--/--/---

**PREPARED BY**

05220000303, Burak Kocabaş

İçindekiler

[1) UNESCO Dünya Mirası Listesi 3](#_Toc150853391)

[1.a UM Alanı Sınıfı Oluşturma 3](#_Toc150853392)

[1.a.1 Kodlar 3](#_Toc150853393)

[1.a.2 Açıklama 3](#_Toc150853394)

[1.b Yedi Elemanlı Bir Generic List Dizisi Oluşturma 3](#_Toc150853395)

[1.b.1 Kodlar 3](#_Toc150853396)

[1.c Dizideki Her Bir Listedeki Bilgilerin ve Her Bir Bölgedeki UM Alanı Sayısını Yazdırma 3](#_Toc150853397)

[1.c.1 Kodlar 3](#_Toc150853398)

[1.c.2 Ekran görüntüleri 3](#_Toc150853399)

[2) YIĞIT ve KUYRUK 3](#_Toc150853400)

[2.a UM Alanı Yığıtı Oluşturma 3](#_Toc150853401)

[2.a.1 Yığıt Sınıfı Kaynak Kodu 3](#_Toc150853402)

[2.a.2 Yığıt Kullanımı Kaynak Kodu 3](#_Toc150853403)

[2.a.3 Ekran Görüntüleri 3](#_Toc150853404)

[2.b UM Alanı Kuyruk Veri Yapısı Oluşturma 3](#_Toc150853405)

[2.b.1 Kuyruk Sınıfı Kaynak Kodu 3](#_Toc150853406)

[2.b.2 Kuyruk Kullanımı Kaynak Kodu 4](#_Toc150853407)

[2.b.3 Ekran Görüntüleri 4](#_Toc150853408)

[3) Öncelikli Kuyruk 4](#_Toc150853409)

[3.a Öncelikli Kuyruk (sınıfı) Oluşturma 4](#_Toc150853410)

[3.a.1 Öncelikli Kuyruk Sınıfı Kaynak Kod 4](#_Toc150853411)

[3.a.2 Öncelikli Kuyruk Kullanımı Kaynak Kodu 4](#_Toc150853412)

[3.a.3 Ekran görüntüleri 4](#_Toc150853413)

[4) Kuyruk ve Öncelikli Kuyruk 4](#_Toc150853414)

[4.a Kuyruk kullanarak süre hesaplanması 4](#_Toc150853415)

[4.a.1 Kaynak Kod 4](#_Toc150853416)

[4.a.2 Ekran görüntüleri 4](#_Toc150853417)

[4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması 4](#_Toc150853418)

[4.b.1 Güncellenmiş Öncelikli Kuyruk Kaynak Kod 4](#_Toc150853419)

[4.b.2 Öncelikli Kuyruk Süre Hesaplaması Kaynak Kod 4](#_Toc150853420)

[4.b.3 Ekran görüntüleri 4](#_Toc150853421)

[4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması 4](#_Toc150853422)

[4.c.1 Kuyruk ve Öncelikli Kuyruk Karşılaştırması 4](#_Toc150853423)

[4.c.2 Önerilen Yöntem 4](#_Toc150853424)

[Öz değerlendirme Tablosu 5](#_Toc150853425)

# 1) UNESCO Dünya Mirası Listesi

//The platform, version, and programming language used

## 1.a UM Alanı Sınıfı Oluşturma

### 1.a.1 Kodlar

|  |
| --- |
| package dataStc2;  import java.util.ArrayList;  public class UM\_Alanı {  private String alanAdi;  private ArrayList<String> ilAdlari;  private int ilanYili;  public UM\_Alanı(String alanAdi, ArrayList<String> ilAdlari, int ilanYili) {  super();  this.alanAdi = alanAdi;  this.ilAdlari = ilAdlari;  this.ilanYili = ilanYili;    }  public String getAlanAdi() {  return alanAdi;  }  public void setAlanAdi(String alanAdi) {  this.alanAdi = alanAdi;  }  public ArrayList<String> getIlAdlari() {  return ilAdlari;  }  public void setIlAdlari(ArrayList<String> ilAdlari) {  this.ilAdlari = new ArrayList<String>(ilAdlari);  }  public int getIlanYili() {  return ilanYili;  }  public void setIlanYili(int ilanYili) {  this.ilanYili = ilanYili;  }      public ArrayList<UM\_Alanı>[] addToRegion(UM\_Alanı object,ArrayList<UM\_Alanı>[] umAlanlarıDizisi) {    if(object.ilAdlari.contains("Antalya")) {    umAlanlarıDizisi[0].add(object);  if(object.ilAdlari.contains("Muğla")) {}  umAlanlarıDizisi[2].add(object);  }  else if(object.ilAdlari.contains("Kars")||object.ilAdlari.contains("Malatya")){  umAlanlarıDizisi[1].add(object);  }  else if (object.ilAdlari.contains("İzmir")||object.ilAdlari.contains("Aydın")||object.ilAdlari.contains("Muğla")||  object.ilAdlari.contains("Denizli") ||object.ilAdlari.contains("Afyon")) {  umAlanlarıDizisi[2].add(object);  }  else if(object.ilAdlari.contains("Adıyaman")||object.ilAdlari.contains("Şanlıurfa")||object.ilAdlari.contains("Diyarbakır")) {  umAlanlarıDizisi[3].add(object);    }  else if(object.ilAdlari.contains("Ankara")||object.ilAdlari.contains("Çorum")||object.ilAdlari.contains("Sivas")  ||object.ilAdlari.contains("Konya")||object.ilAdlari.contains("Nevşehir")||object.ilAdlari.contains("Eskişehir")) {  umAlanlarıDizisi[4].add(object);    }  else if(object.ilAdlari.contains("Karabük")|| object.ilAdlari.contains("Kastamonu")) {  umAlanlarıDizisi[5].add(object);  }  else if(object.ilAdlari.contains("İstanbul")||object.ilAdlari.contains("Çanakkkale")||object.ilAdlari.contains("Edirne")||object.ilAdlari.contains("Bursa")) {  umAlanlarıDizisi[6].add(object);  }  return umAlanlarıDizisi;      }              } |

### 1.a.2 Açıklama

//Kullanılan veri yapılarını/ veri tiplerinin kısaca anlatımını burada gerçekleştiriniz

## 1.b Yedi Elemanlı Bir Generic List Dizisi Oluşturma

### 1.b.1 Kodlar

|  |
| --- |
| package dataStc2;  import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  public class UNESCOExample {    public static ArrayList<UM\_Alanı>[] returnListMetod(){  String unescoVerileri = "Divriği Ulu Camii ve Darüşşifası (Sivas) 1985\n" +  "İstanbul'un Tarihi Alanları (İstanbul) 1985\n" +  "Göreme Millî Parkı ve Kapadokya (Nevşehir) 1985 (Karma Miras Alanı)\n" +  "Hattuşa: Hitit Başkenti (Çorum) 1986\n" +  "Nemrut Dağı (Adıyaman) 1987\n" +  "Hieropolis-Pamukkale (Denizli) 1988 (Karma Miras Alanı)\n" +  "Xanthos-Letoon (Antalya-Muğla) 1988\n" +  "Safranbolu Şehri (Karabük) 1994\n" +  "Truva Arkeolojik Alanı (Çanakkale) 1998\n" +  "Edirne Selimiye Camii ve Külliyesi (Edirne) 2011\n" +  "Çatalhöyük Neolitik Alanı (Konya) 2012\n" +  "Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu (Bursa) 2014\n" +  "Bergama Çok Katmanlı Kültürel Peyzaj Alanı (İzmir) 2014\n" +  "Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel Peyzajı (Diyarbakır) 2015\n" +  "Efes (İzmir) 2015\n" +  "Ani Arkeolojik Alanı (Kars) 2016\n" +  "Aphrodisias (Aydın) 2017\n" +  "Göbekli Tepe (Şanlıurfa) 2018\n" +  "Arslantepe Höyüğü (Malatya) 2021\n" +  "Gordion (Ankara) 2023\n"+  "Eşrefoğlu Camii (Konya) 2023\n"+  "Mahmut Bey Camii (Kastamonu) 2023\n"+  "Sivrihisar Camii (Eskişehir) 2023\n"+  "Afyon Ulu Camii (Afyon) 2023\n"+  "Arslanhane Camii (Ankara) 2023";      String[] satirlar = unescoVerileri.split("\n");    String [] bolgelerListesi = {"Akdeniz","Doğu Anadolu","Ege","Güney Doğu Anadolu","İç Anadolu","Karadeniz","Marmara"};    ArrayList<UM\_Alanı> [] umAlanlarıDizisi = new ArrayList[7];      if (umAlanlarıDizisi[0] == null) {  for (int i = 0; i < 7; i++) {  umAlanlarıDizisi[i] = new ArrayList<>();  }  }    for (String satir : satirlar) {    String[] parcalar = satir.split("\\(|\\)");    String alanAdi = parcalar[0].trim();    String[] iller = parcalar[1].trim().split("-");  List<String> ilListesi = Arrays.asList(iller);    int ilanYili = Integer.parseInt(parcalar[2].trim());    UM\_Alanı umAlan = new UM\_Alanı(alanAdi, new ArrayList<>(ilListesi), ilanYili);  umAlanlarıDizisi= umAlan.addToRegion(umAlan,umAlanlarıDizisi);    }  return umAlanlarıDizisi;  }  } |

## 1.c Dizideki Her Bir Listedeki Bilgilerin ve Her Bir Bölgedeki UM Alanı Sayısını Yazdırma

### 1.c.1 Kodlar

|  |
| --- |
| ArrayList<UM\_Alanı>[] umAlanlarıDizisi = new ArrayList[7];  umAlanlarıDizisi = UNESCOExample.returnListMetod();  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  for (UM\_Alanı umAlan : list) {  System.out.print(umAlan.getAlanAdi() + " - " + umAlan.getIlAdlari() + " - " + umAlan.getIlanYili()+" ");      }  System.out.println("/// Bölgedeki Unesco Miran Alanı Sayısı: " + list.size());  System.out.println(""); |

### 1.c.2 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz

metin, ekran görüntüsü, yazı tipi, bilgi, enformasyon içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 2) YIĞIT ve KUYRUK

//The platform, version, and programming language used

## 2.a UM Alanı Yığıtı Oluşturma

### 2.a.1 Yığıt Sınıfı Kaynak Kodu

|  |
| --- |
| package dataStc2;  import java.util.ArrayList;  public class Stack {        private int maxSize;  private UM\_Alanı[] stackArray;  private int top;      public Stack() {  this.maxSize = maxSize;  this.stackArray = stackArray;  this.top = top;  }  public Stack(int max) //  {  maxSize = max;  stackArray = new UM\_Alanı[maxSize];  top = -1;  }      public void push(UM\_Alanı object) // put item on top of stack  {  stackArray[++top] = object;  }      public UM\_Alanı pop() // take item from top of stack  {  return stackArray[top--];  }  //    public UM\_Alanı peek() // peek at top of stack  {  return stackArray[top];  }      public boolean isEmpty() // true if stack is empty  {  return (top == -1);  }  public void popAndPrintStack() {  UM\_Alanı temp;  int totalSize = 0;    ArrayList<UM\_Alanı>[] umAlanlarıDizisi = new ArrayList[7];  umAlanlarıDizisi = UNESCOExample.returnListMetod();  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  totalSize += list.size();  }  Stack stackList = new Stack(totalSize);  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  for (UM\_Alanı umAlan : list) {  stackList.push(umAlan);  }  }  while(!stackList.isEmpty()){  temp = stackList.peek();  stackList.pop();  System.out.println("Listeden Son Çıkarlıan UM ALANI : " + temp.getAlanAdi() + " - " + temp.getIlAdlari() + " - " + temp.getIlanYili()+" ");    }    }    } |

### 2.a.2 Yığıt Kullanımı Kaynak Kodu

|  |
| --- |
| Stack stack = new Stack();  stack.popAndPrintStack(); |

### 2.a.3 Ekran Görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz

metin, ekran görüntüsü, siyah beyaz, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 2.b UM Alanı Kuyruk Veri Yapısı Oluşturma

### 2.b.1 Kuyruk Sınıfı Kaynak Kodu

|  |
| --- |
| dataStc2;  import java.util.ArrayList;  public class Queue {    private int maxSize;  private UM\_Alanı[] queArray;  private int front;  private int rear;  private int nItems;    public Queue() {  super();  this.maxSize = maxSize;  this.queArray = queArray;  this.front = front;  this.rear = rear;  this.nItems = nItems;  }  public Queue(int s)  {  maxSize = s;  queArray = new UM\_Alanı[maxSize];  front = 0;  rear = -1;  nItems = 0;  }      public void insert(UM\_Alanı object)  {  if(rear == maxSize-1)  rear = -1;  queArray[++rear] = object;    nItems++;  }      public UM\_Alanı remove() {  UM\_Alanı temp = queArray[front++];  if(front == maxSize)  front = 0;  nItems--;  return temp;  }      public UM\_Alanı peekFront()  {  return queArray[front];  }      public boolean isEmpty()  {  return (nItems==0);  }      public boolean isFull()  {  return (nItems==maxSize);  }    public int size()  {  return nItems;  }  public void popAndPrintQueue() {  UM\_Alanı temp;  int totalSize = 0;    ArrayList<UM\_Alanı>[] umAlanlarıDizisi = new ArrayList[7];  umAlanlarıDizisi = UNESCOExample.returnListMetod();  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  totalSize += list.size();  }  Queue queueList = new Queue(totalSize);  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  for (UM\_Alanı umAlan : list) {  queueList.insert(umAlan);  }  }  while(!queueList.isEmpty()){  temp = queueList.peekFront();  queueList.remove();  System.out.println("Listenin Başından Çıkarlıan UM ALANI : " + temp.getAlanAdi() + " - " + temp.getIlAdlari() + " - " + temp.getIlanYili()+" ");    }    }        } |

### 2.b.2 Kuyruk Kullanımı Kaynak Kodu

|  |
| --- |
|  |

### 2.b.3 Ekran Görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyinizmetin, ekran görüntüsü, kalıp, desen, düzen, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 3) Öncelikli Kuyruk

## 3.a Öncelikli Kuyruk (sınıfı) Oluşturma

### 3.a.1 Öncelikli Kuyruk Sınıfı Kaynak Kod

|  |
| --- |
| package dataStc2;  import java.util.ArrayList;  public class Queue {    private int maxSize;  private UM\_Alanı[] queArray;  private int front;  private int rear;  private int nItems;    public Queue() {  super();  this.maxSize = maxSize;  this.queArray = queArray;  this.front = front;  this.rear = rear;  this.nItems = nItems;  }  public Queue(int s)  {  maxSize = s;  queArray = new UM\_Alanı[maxSize];  front = 0;  rear = -1;  nItems = 0;  }      public void insert(UM\_Alanı object)  {  if(rear == maxSize-1)  rear = -1;  queArray[++rear] = object;    nItems++;  }      public UM\_Alanı remove() {  UM\_Alanı temp = queArray[front++];  if(front == maxSize)  front = 0;  nItems--;  return temp;  }      public UM\_Alanı peekFront()  {  return queArray[front];  }      public boolean isEmpty()  {  return (nItems==0);  }      public boolean isFull()  {  return (nItems==maxSize);  }    public int size()  {  return nItems;  }  public void popAndPrintQueue() {  UM\_Alanı temp;  int totalSize = 0;    ArrayList<UM\_Alanı>[] umAlanlarıDizisi = new ArrayList[7];  umAlanlarıDizisi = UNESCOExample.returnListMetod();  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  totalSize += list.size();  }  Queue queueList = new Queue(totalSize);  for (ArrayList<UM\_Alanı> list : umAlanlarıDizisi) {  for (UM\_Alanı umAlan : list) {  queueList.insert(umAlan);  }  }  while(!queueList.isEmpty()){  temp = queueList.peekFront();  queueList.remove();  System.out.println("Listenin Başından Çıkarlıan UM ALANI : " + temp.getAlanAdi() + " - " + temp.getIlAdlari() + " - " + temp.getIlanYili()+" ");    }    }        } |

### 3.a.2 Öncelikli Kuyruk Kullanımı Kaynak Kodu

|  |
| --- |
| Queue queue = new Queue();  queue.popAndPrintQueue(); |

### 3.a.3 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz metin, ekran görüntüsü, yazı tipi, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturulduan

# 4) Kuyruk ve Öncelikli Kuyruk

## 4.a Kuyruk kullanarak süre hesaplanması

### 4.a.1 Kaynak Kod

|  |
| --- |
| package dataStc2;  public class MarketQueue {  private int [] productCountList = {10, 4, 8, 6, 7, 1, 15, 9, 3, 2};  private int maxSize;  private int[] queArray;  private int front;  private int rear;  private int nItems;  public MarketQueue() {  super();  this.productCountList = productCountList;  this.maxSize = maxSize;  this.queArray = queArray;  this.front = front;  this.rear = rear;  this.nItems = nItems;  }    public MarketQueue(int s) // constructor  {  maxSize = s;  queArray = new int[maxSize];  front = 0;  rear = -1;  nItems = 0;  }      public void insert(int j)  {  if(rear == maxSize-1)  rear = -1;  queArray[++rear] = j;  nItems++;  }      public int remove()  {  int temp = queArray[front++];  if(front == maxSize)  front = 0;  nItems--;  return temp;  }    public int peekFront() {  return queArray[front];  }    public boolean isEmpty()  {  return (nItems==0);  }    public boolean isFull()  {  return (nItems==maxSize);  }    public int size()  {  return nItems;    }  public void calculatingCustomers() {  int temp;  double tempTime;  double customersWaitingTime= 0;  double allTime =0;  MarketQueue queueList= new MarketQueue(10);  for (int i:productCountList) {  queueList.insert(i);;  }  while(!queueList.isEmpty()) {  temp=queueList.peekFront();  tempTime=(temp \*2.5);  customersWaitingTime += tempTime;  allTime += customersWaitingTime;  queueList.remove();  System.out.println((productCountList.length - queueList.nItems)+". Müşterinin beklediği süre : " + customersWaitingTime);  }  System.out.println("Bu kuyruğun müşterileri ortalama beklettiği süre: " + allTime/10);    }            } |

### 4.a.2 Ekran görüntüleri

## metin, ekran görüntüsü, yazı tipi, siyah içeren bir resim Açıklama otomatik olarak oluşturuldu4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması

### 4.b.1 Güncellenmiş Öncelikli Kuyruk Kaynak Kod

|  |
| --- |
| package dataStc2;  import java.util.LinkedList;  import java.util.List;  public class PriorityQueueCopy {  private int[] productCountList = {10, 4, 8, 6, 7, 1, 15, 9, 3, 2};  private List<Integer> queArray;  private int nItems ;    public PriorityQueueCopy()  {  queArray = new LinkedList<>();  nItems = 0;  }  public void insert(int i) {  queArray.add(i);  nItems++;    }  public void calculatingCustomersPriorityQueue (PriorityQueueCopy prioCopy){  for(int i:productCountList) {  prioCopy.insert(i);  }  int temp;  double tempTime;  double customersWaitingTime= 0;  double allTime =0;  int compareToValue ;    while(!prioCopy.isEmpty()){  int tempIndex = 0;    if(queArray.size()>0) {  for(int i=0;i<queArray.size()-1;i++) {  compareToValue = queArray.get(tempIndex).compareTo(queArray.get(i+1));  if(compareToValue<0) {  tempIndex = tempIndex;  }  else {  tempIndex = i+1;  }    }  temp= queArray.get(tempIndex);  tempTime = 2.5\*temp;  customersWaitingTime+= tempTime;  allTime += customersWaitingTime;  queArray.remove(tempIndex);  nItems--;  System.out.println((productCountList.length-prioCopy.nItems)+". Müşterinin Beklediği Süre : "+ customersWaitingTime);    }    }  System.out.println("Müşterilerin tümü için öncelikli kuyruktaki ortalama bekleme süresi : " + (allTime/10));    }  public boolean isEmpty()  {  return (nItems==0);  }        public int size()  {  return nItems;  }        } |

### 4.b.2 Öncelikli Kuyruk Süre Hesaplaması Kaynak Kod

|  |
| --- |
| public void calculatingCustomersPriorityQueue (PriorityQueueCopy prioCopy){  for(int i:productCountList) {  prioCopy.insert(i);  }  int temp;  double tempTime;  double customersWaitingTime= 0;  double allTime =0;  int compareToValue ;    while(!prioCopy.isEmpty()){  int tempIndex = 0;    if(queArray.size()>0) {  for(int i=0;i<queArray.size()-1;i++) {  compareToValue = queArray.get(tempIndex).compareTo(queArray.get(i+1));  if(compareToValue<0) {  tempIndex = tempIndex;  }  else {  tempIndex = i+1;  }    }  temp= queArray.get(tempIndex);  tempTime = 2.5\*temp;  customersWaitingTime+= tempTime;  allTime += customersWaitingTime;  queArray.remove(tempIndex);  nItems--;  System.out.println((productCountList.length-prioCopy.nItems)+". Müşterinin Beklediği Süre : "+ customersWaitingTime);    }    }  System.out.println("Müşterilerin tümü için öncelikli kuyruktaki ortalama bekleme süresi : " + (allTime/10));    } |

### 4.b.3 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyinizmetin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması

### 4.c.1 Kuyruk ve Öncelikli Kuyruk Karşılaştırması

PQ ortalama süre bakımından Queue dan çok daha verimli gözükse de herkes için aynı adiliyet söz konusu değil İlk gelmenin hiçbir önemi olmadığından sepetinde çok malzeme olan müşteri queue ye göre oldukça fazla bekleyebiliyor.Queue da ise sadece 1 malzeme bile almak isterseniz en son geldiyseniz önünüzde oldukça fazla müşteri olduğundan çok beklemek durumunda kalıyorsunuz ve bu bekleme PQ nun sonucusuna göre oldukça fazla bir bekleme.

### 4.c.2 Önerilen Yöntem

Bulunabiliecek optimal bir k Sabiti ile hem sırasını hem de elindeki malzeme sayısını çarpıp toplarsak yeni bir ağırlık elde ederiz ve müşterileri bu ağırlık sırasına göre sıralarsak daha verimli bir kuyruk elde edilebilir

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) Sınıf Oluşturma** | **5** | **5** | **Yapıldı** |
| **1 b) Bileşik veri yapısının oluşturulması ve elemanların eklenmesi; Kaynak kod, ekran görüntüsü.** | **15** | **12** | **Eleman ekleme işleminde if statementları çok kötü duruyor daha mantıklı ve az uğraştırıcı şekilde yapılabilir** |
| **1 c) Veri yapısının elemanlarının listelenmesi ve her bölgedeki UM alanı sayısının yazdırılması; Kaynak kodlar, ekran görüntüsü.** | **10** | **10** | **Her şeyi doğru şekilde yazdırıyor** |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **10** | **10** | **İşlemleri doğru yapıyor** |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **10** | **10** | **İşlemleri Doğru yapıyor** |
| **3 Öncelikli Kuyruk kod ve ekran görüntüleri** | **20** | **18** | **Çatalhöyük olan bir UM alanı ismini Ç den dolayı alfabetik sıralayamıyor en son onu çıkarıyor** |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri, karşılaştırma ve soruların cevapları.** | **20** | **17** | **Kodum çok sade gözükmüyor daha temiz yazılabilir** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** |  |
| **Toplam** | **100** | **92** |  |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.**

**Not: Raporu teslim edilmeyen projeler değerlendirmeye alınmayacaktır.**