

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2022–2023 FALL SEMESTER**

**PROJECT-2 REPORT**

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

**DELIVERY DATE**

05/12/2022

**PREPARED BY**

05210000281, Salih Ali Kamacı

İçindekiler

[1) Türkiye’deki Milli Parklar Listesi 3](#_Toc118965473)

[1.a Milli Park Sınıfı Oluşturma 3](#_Toc118965474)

[1.a.1 Kodlar 3](#_Toc118965475)

[1.a.2 Açıklama 3](#_Toc118965476)

[1.b İki Elemanlı Bir Generic List Dizisi Oluşturma 3](#_Toc118965477)

[1.b.1 Kodlar 3](#_Toc118965478)

[1.c Dizideki Her Bir Listedeki Bilgilerin ve Elemanların Yüzölçümlerinin Toplamının Yazdırılması 3](#_Toc118965479)

[1.c.1 Kodlar 4](#_Toc118965480)

[1.c.2 Ekran görüntüleri 4](#_Toc118965481)

[2) YIĞIT ve KUYRUK 7](#_Toc118965482)

[2.a Milli Park Yığıtı Oluşturma 9](#_Toc118965483)

[2.a.1 Kaynak Kod 9](#_Toc118965484)

[2.a.2 Ekran Görüntüleri 9](#_Toc118965485)

[2.b Milli Park Kuyruk Veri Yapısı Oluşturma 12](#_Toc118965486)

[2.b.1 Kaynak Kod 12](#_Toc118965487)

[2.b.2 Ekran Görüntüleri 12](#_Toc118965488)

[3) Öncelikli Kuyruk 15](#_Toc118965489)

[3.a Öncelikli Kuyruk (sınıfı) Oluşturma 15](#_Toc118965490)

[3.a.1 Kaynak Kod 15](#_Toc118965491)

[3.a.2 Ekran görüntüleri 17](#_Toc118965492)

[4) Kuyruk ve Öncelikli Kuyruk 19](#_Toc118965493)

[4.a Kuyruk kullanarak süre hesaplanması 19](#_Toc118965494)

[4.a.1 Kaynak Kod 19](#_Toc118965495)

[4.a.2 Ekran görüntüleri 19](#_Toc118965496)

[4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması 21](#_Toc118965497)

[4.b.1 Kaynak Kod 21](#_Toc118965498)

[4.b.2 Ekran görüntüleri 22](#_Toc118965499)

[4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması 22](#_Toc118965500)

[Öz değerlendirme Tablosu 22](#_Toc118965501)

# 1) Türkiye’deki Milli Parklar Listesi

## 1.a Milli Park Sınıfı Oluşturma

### 1.a.1 Kodlar

private string park\_adi; //Millipark nesnesinin özelliklerini tutan değişkenler

private string il\_adi;

private int park\_ilan\_yili;

private double park\_yuzolcumu;

static double toplamkucuk = 0 ;

static double toplambuyuk = 0;

} //Private tipi değişkenler için verilere ulaşılması adına get metodları

public string Park\_adi { get { return park\_adi; } }

public string İl\_adi { get { return il\_adi; } }

public int Park\_ilan\_yili { get { return park\_ilan\_yili; } }

public double Park\_yuzolcumu { get { return park\_yuzolcumu; } }

public milliPark(string ad, string il\_ad, int ilan\_yili, double yuzolcumu) //Parametleri constructor

{

park\_adi = ad;

il\_adi = il\_ad;

park\_ilan\_yili = ilan\_yili;

park\_yuzolcumu = yuzolcumu;}

### 1.a.2 Açıklama

## park\_adi = Parkın adını tutan constructorda ad değişkeni ile atanan değer.

## il\_adi = Parkın bulunduğu ili tutan constructorda il\_ad değişkeni ile atanan değer.

## park\_ilan\_yili = Parkın ilan edildiği yılı tutan constructorda ilan\_yili değişkeni ile atanan değer.

## park\_yuzolcumu= Parkın yüzölçümünü tutan constructorda yuzolcumu değişkeni ile atanan değer.

## Park\_adi = park\_adi değerinin get metodu.

## İl\_adi = il\_adi değerinin get metodu.

## Park\_ilan\_yili = park\_ilan\_yili değerinin get metodu.

## Park\_yuzolcumu = park\_yuzolcumu değerinin get metodu.

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

### 1.b.1 Kodlar

List<milliPark> kucukmp = new List<milliPark>(); //15.000 den küçük yüzölçümüne sahip parklar kucukmp ye atanır.

List<milliPark> buyukmp = new List<milliPark>(); //15.000 den büyük yüzölçümüne sahip parklar buyukmp ye atanır.

object[] yuzolcumluListe = new object[2] {kucukmp,buyukmp};

## 1.c Dizideki Her Bir Listedeki Bilgilerin ve Elemanların Yüzölçümlerinin Toplamının Yazdırılması

### 1.c.1 Kodlar

if (milliParklar.Park\_yuzolcumu < 15.000)

{

toplamkucuk += milliParklar.Park\_yuzolcumu;

kucukmp.Add(milliParklar);}

else

{

toplambuyuk += milliParklar.Park\_yuzolcumu;

buyukmp.Add(milliParklar);}

for (int i = 0; i < kucukmp.Count; i++)

{ Console.WriteLine("YÜZÖLÇÜMÜ 15.000`DEN KÜÇÜK PARK");

Console.WriteLine(" ");

Yazdir(kucukmp[i]); }

for (int i = 0; i < buyukmp.Count; i++)

{ Console.WriteLine("YÜZÖLÇÜMÜ 15.000`DEN BÜYÜK PARK");

Console.WriteLine(" ");

Yazdir(buyukmp[i]); }

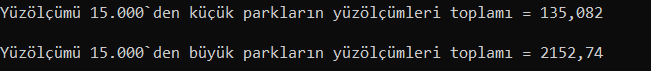
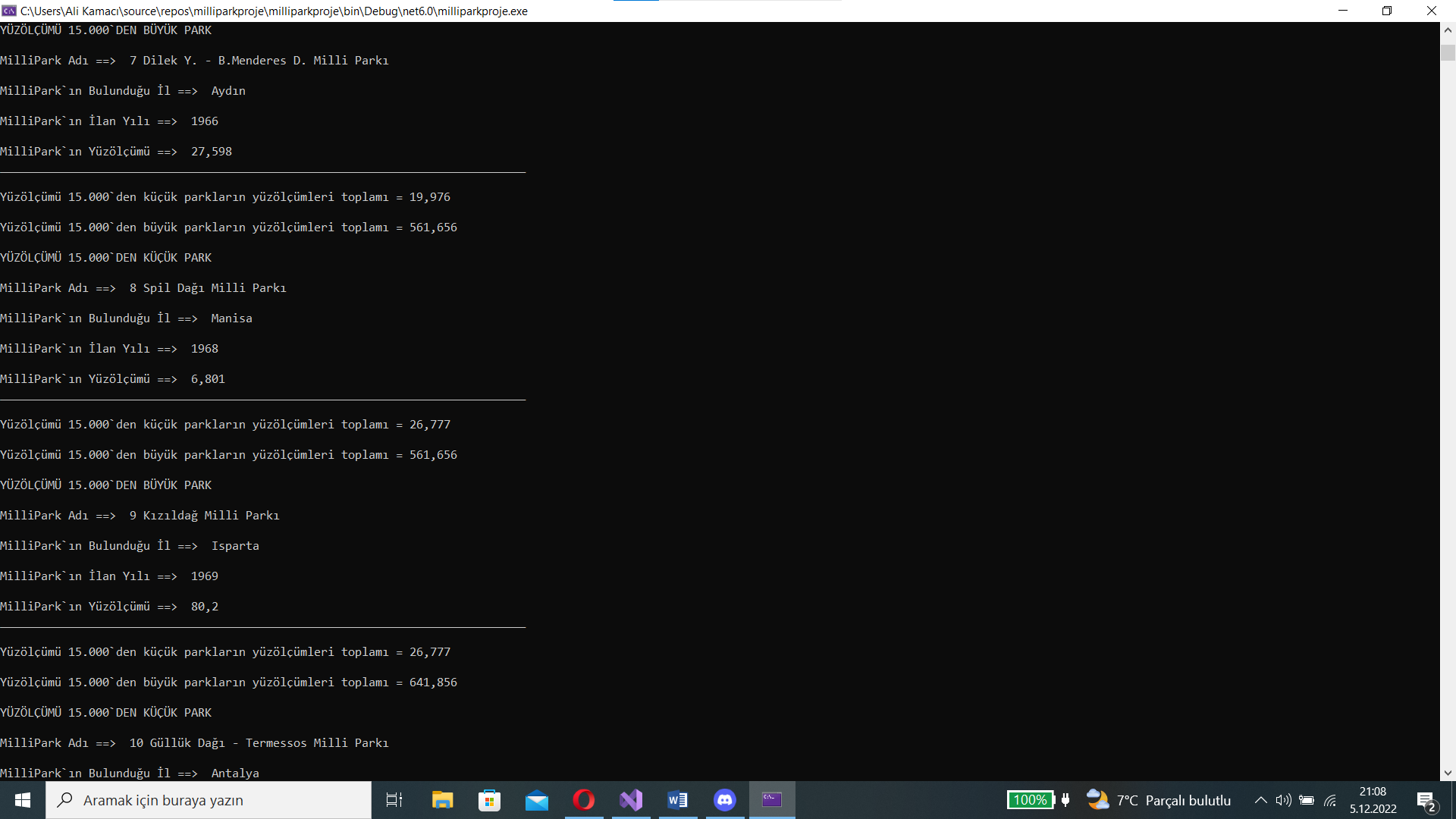
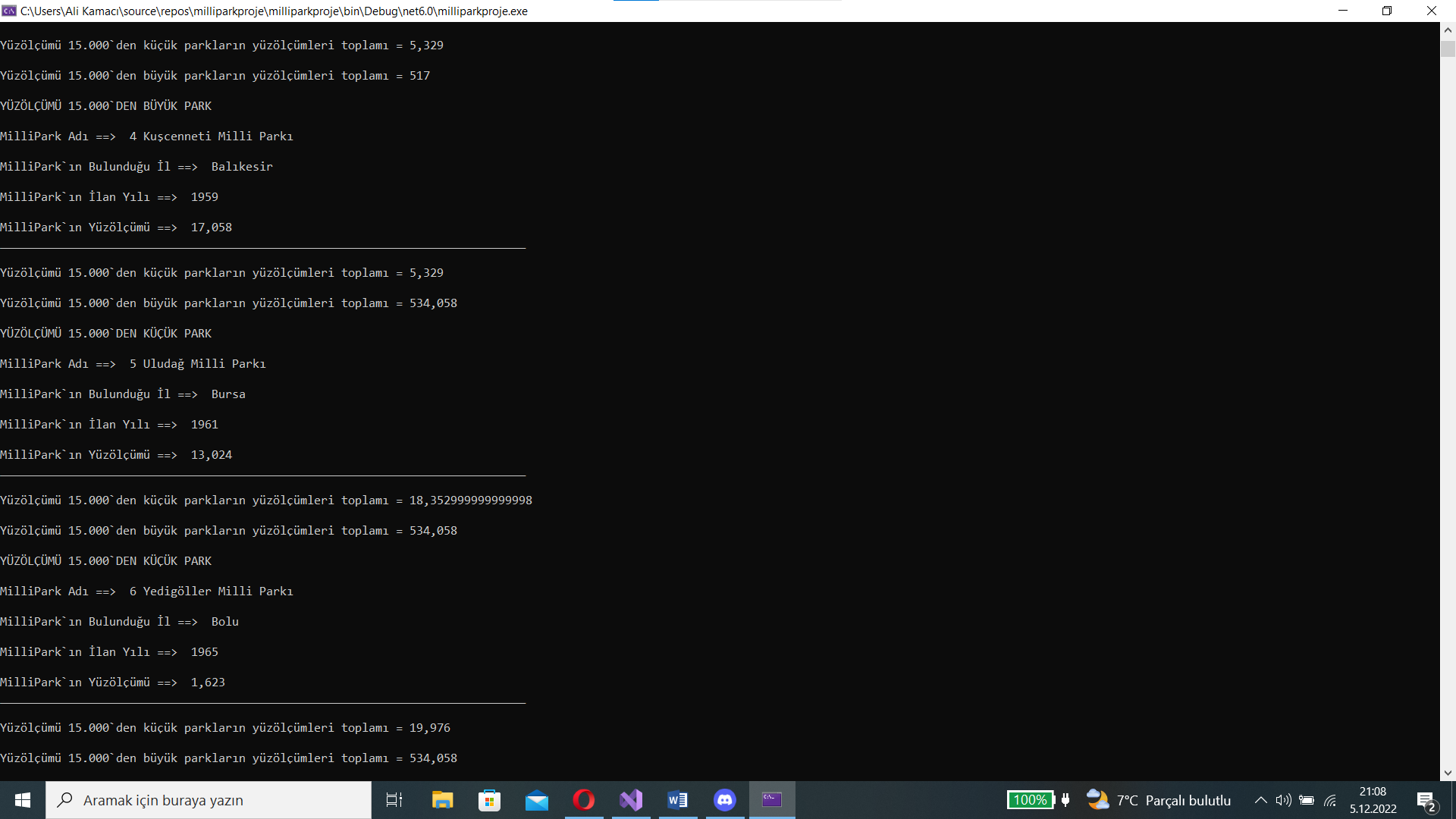
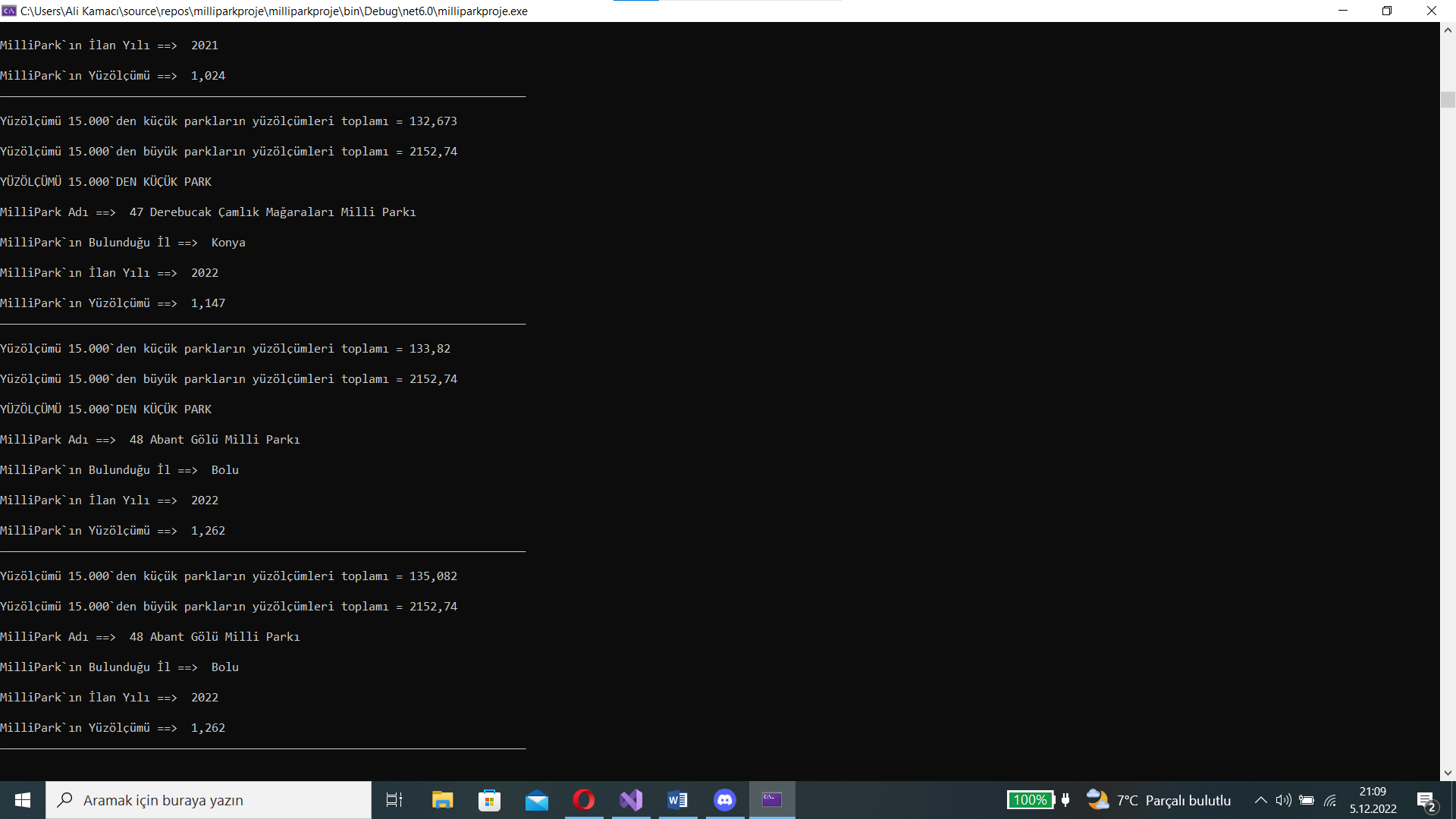
Console.WriteLine("Yüzölçümü 15.000`den küçük parkların yüzölçümleri toplamı = {0}",toplamkucuk);

Console.WriteLine(" ");

Console.WriteLine("Yüzölçümü 15.000`den büyük parkların yüzölçümleri toplamı = {0}",toplambuyuk);Console.WriteLine(" ");

### 1.c.2 Ekran görüntüleri

# C:\Users\Ali Kamacı\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.png



# 2) YIĞIT ve KUYRUK

//The platform, version, and programming language used

## 2.a Milli Park Yığıtı Oluşturma

### 2.a.1 Kaynak Kod

private int maxSize; // size of stack array

private milliPark[] stackArray;

private int top; // top of stack

public Stack(int s) // constructor

{

maxSize = s; // set array size

stackArray = new milliPark[maxSize]; // create array

top = -1; // no items yet

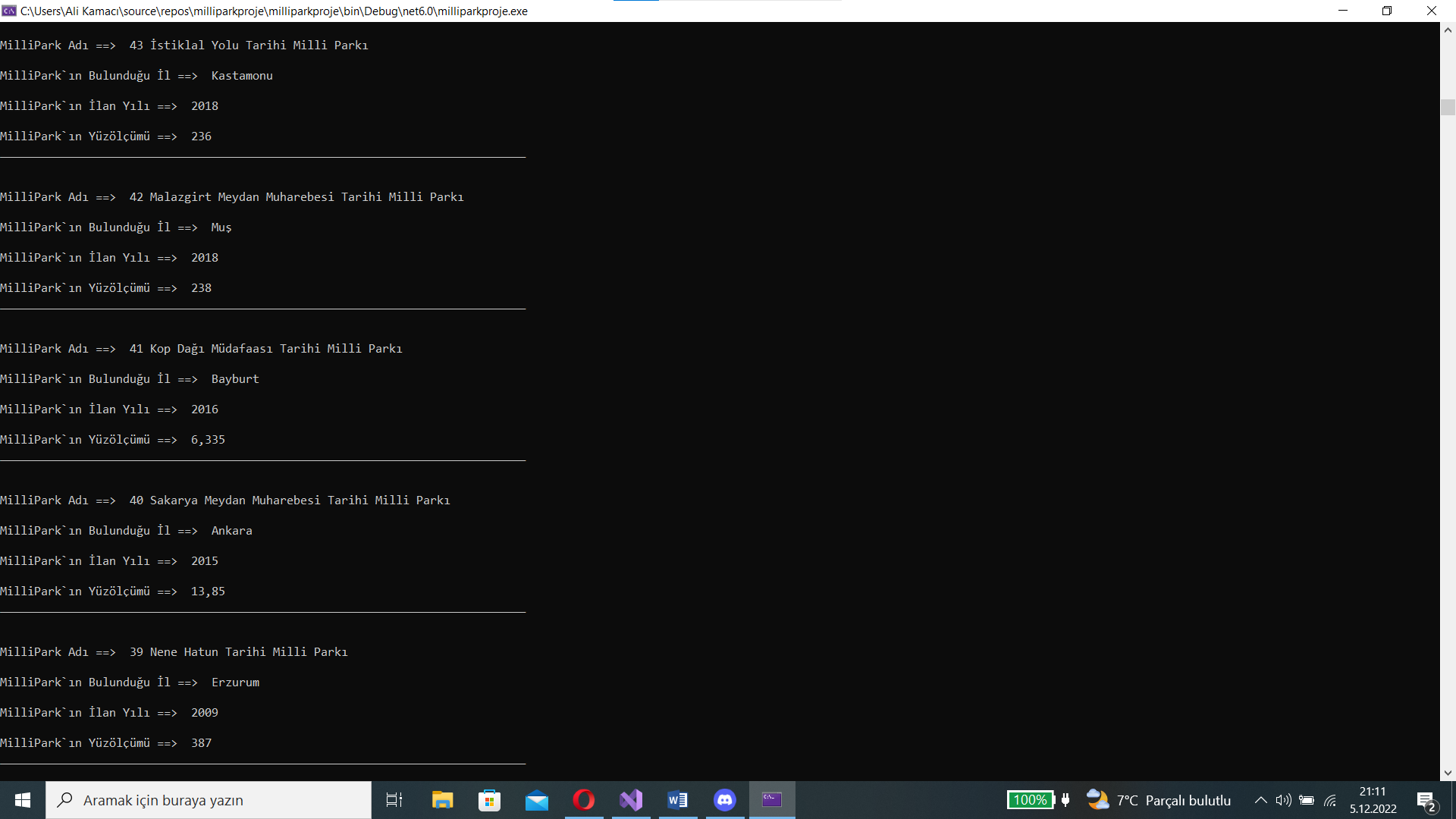
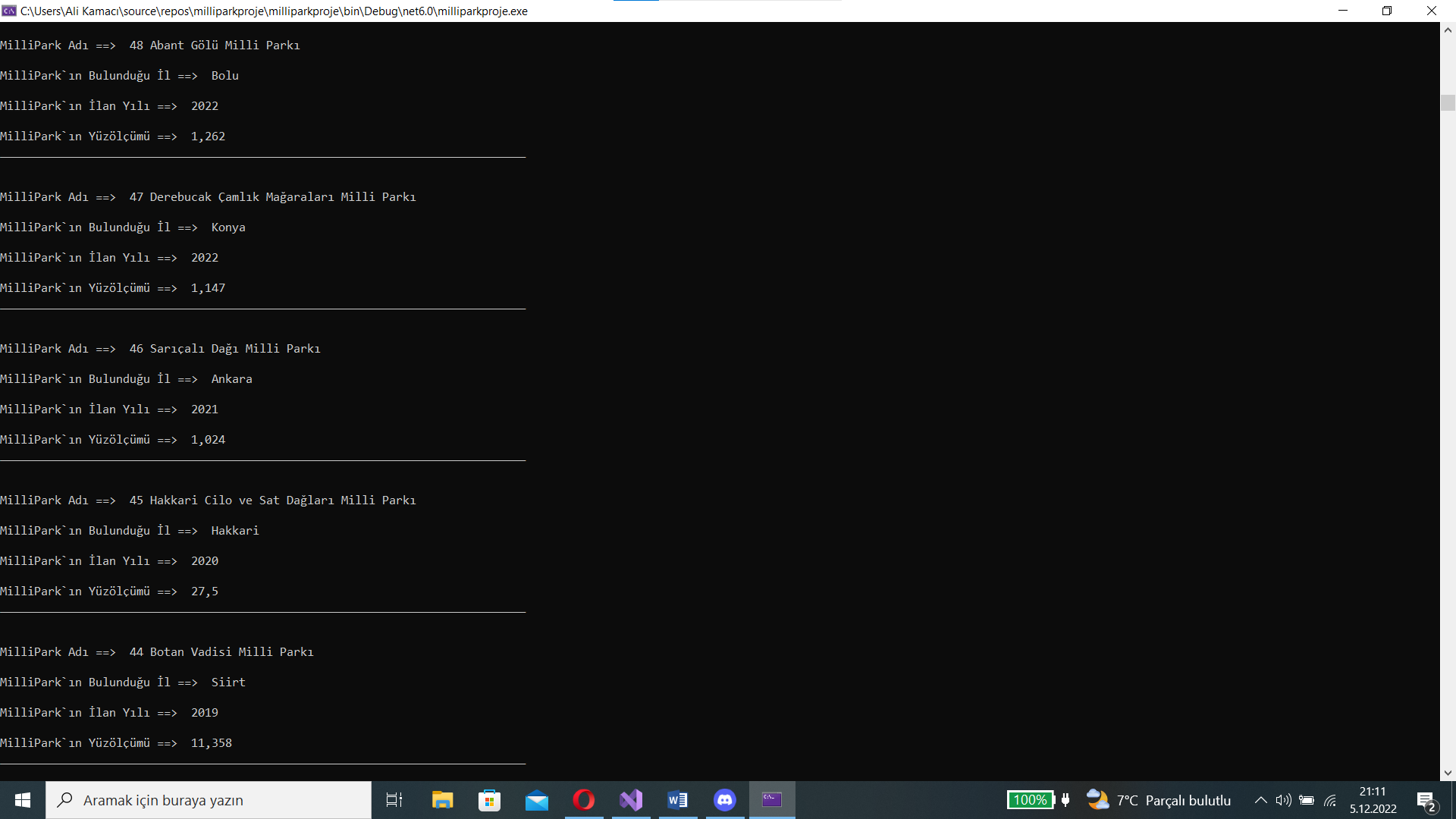
}

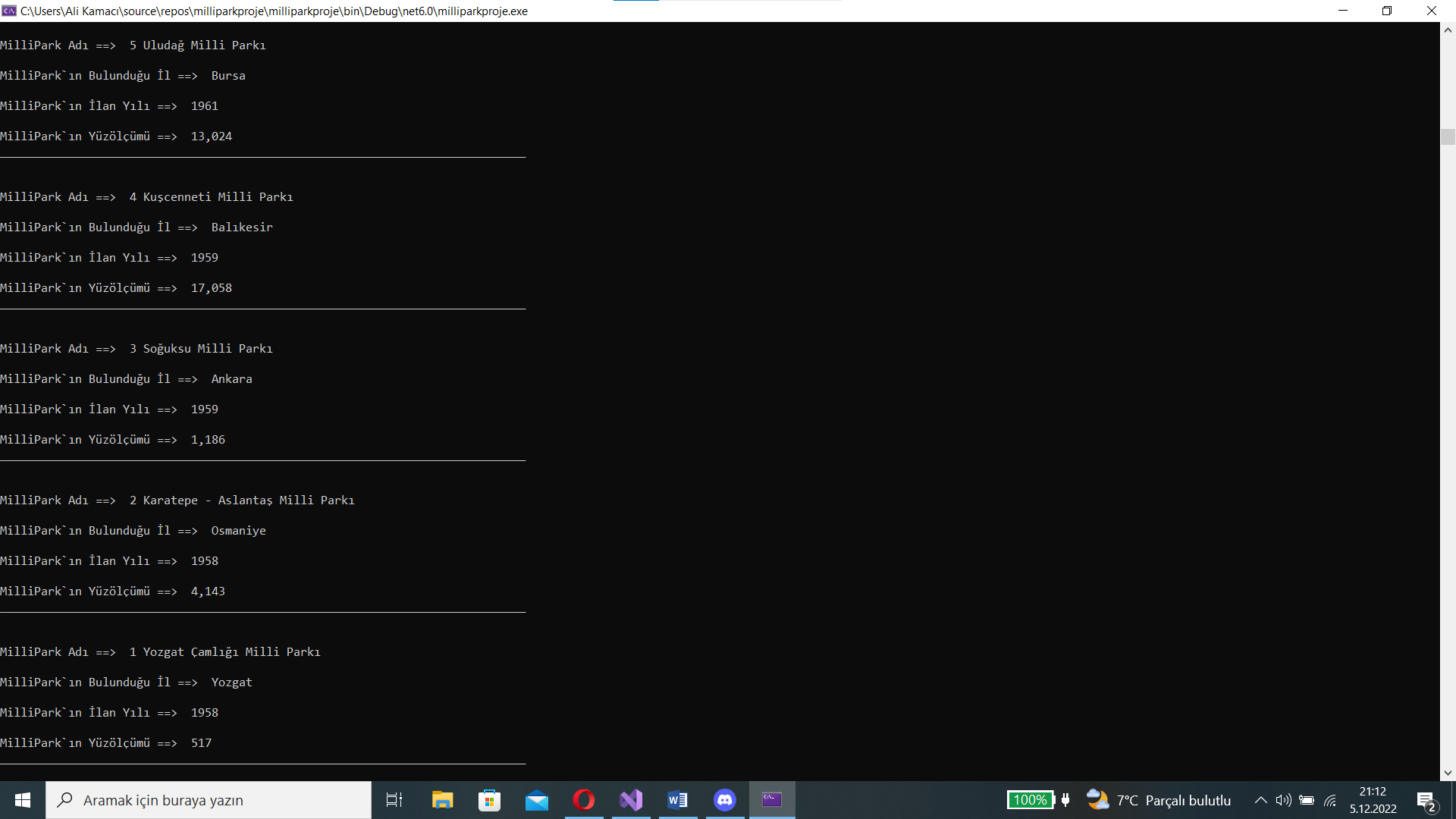
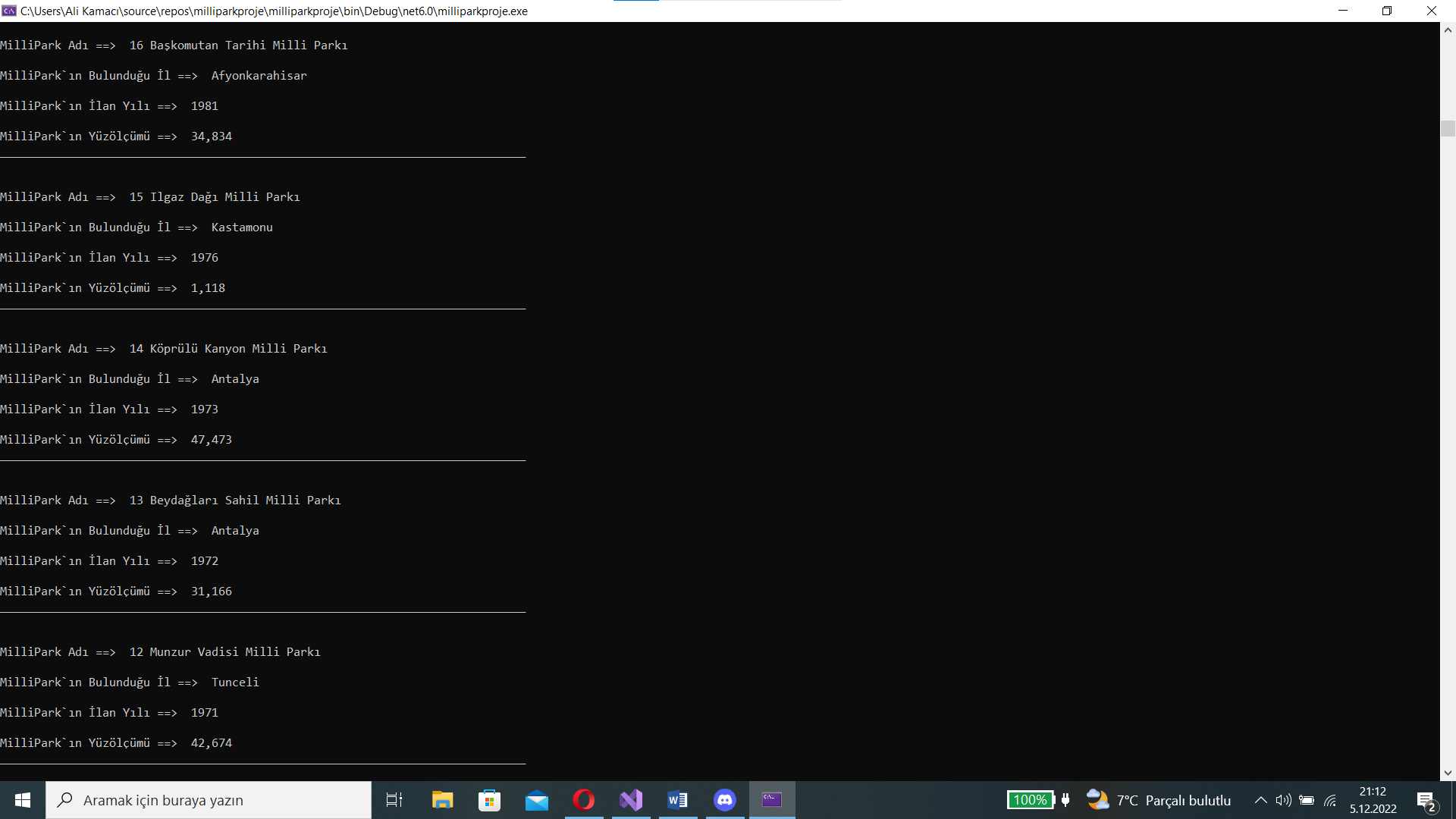
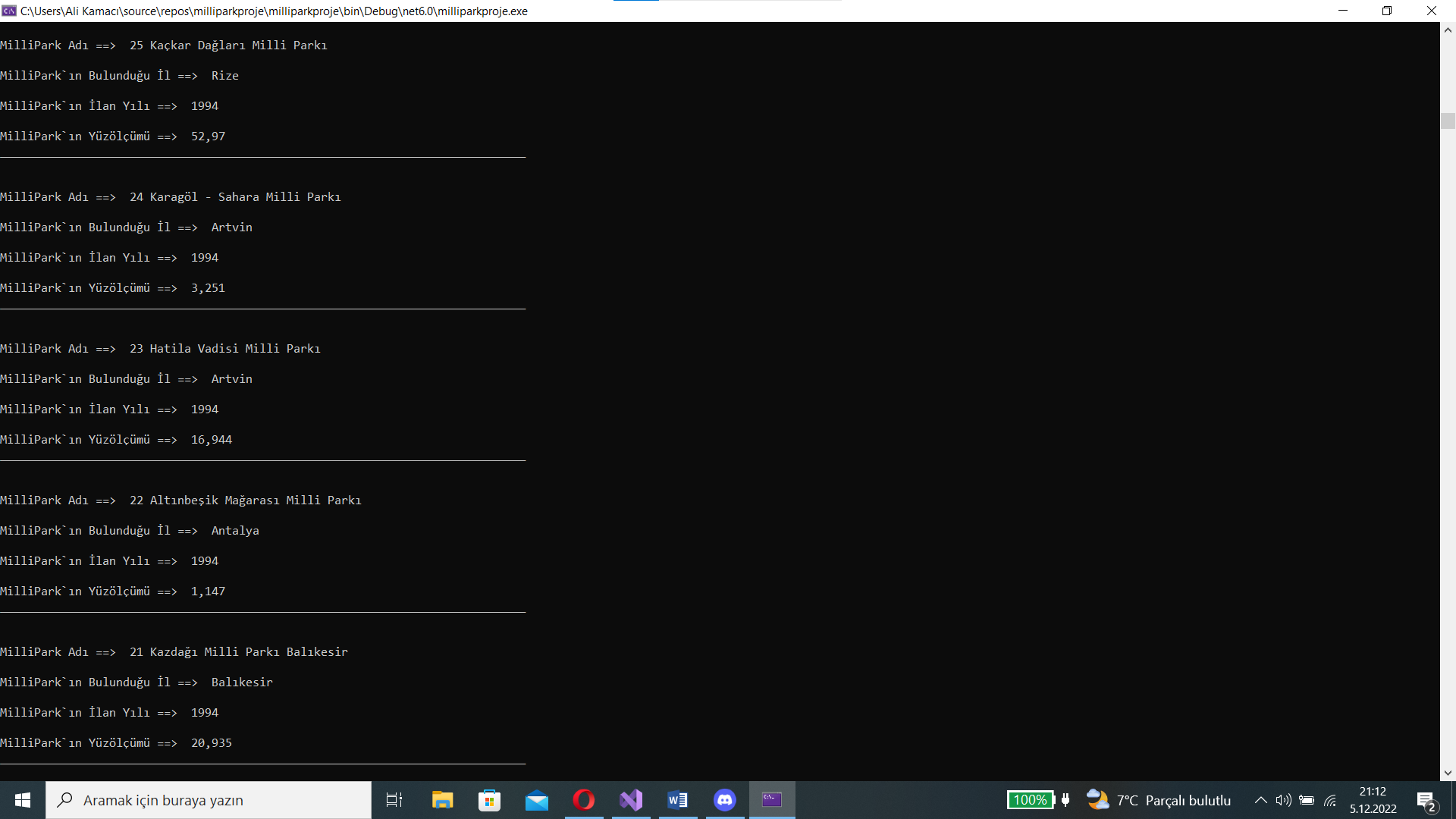
public void push(milliPark j) // put item on top of stack

{

stackArray[++top] = j; // increment top, insert item}

### 2.a.2 Ekran Görüntüleri





## 2.b Milli Park Kuyruk Veri Yapısı Oluşturma

### 2.b.1 Kaynak Kod

private int maxSize; //queArray in maksimum alabileceği boyut

private milliPark[] queArray; //Millipark nesnelerini kuyruk için tutan array

private int front; //Arrayin başını tutan değişken

private int rear; //Arrayin sonunu tutan değişken

private int nItems; //Arrayin item sayısını tutan değişken

public Queue(int s) // constructor

{

maxSize = s;

queArray = new milliPark[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

public void insert(milliPark j) // put item at rear of queue

{

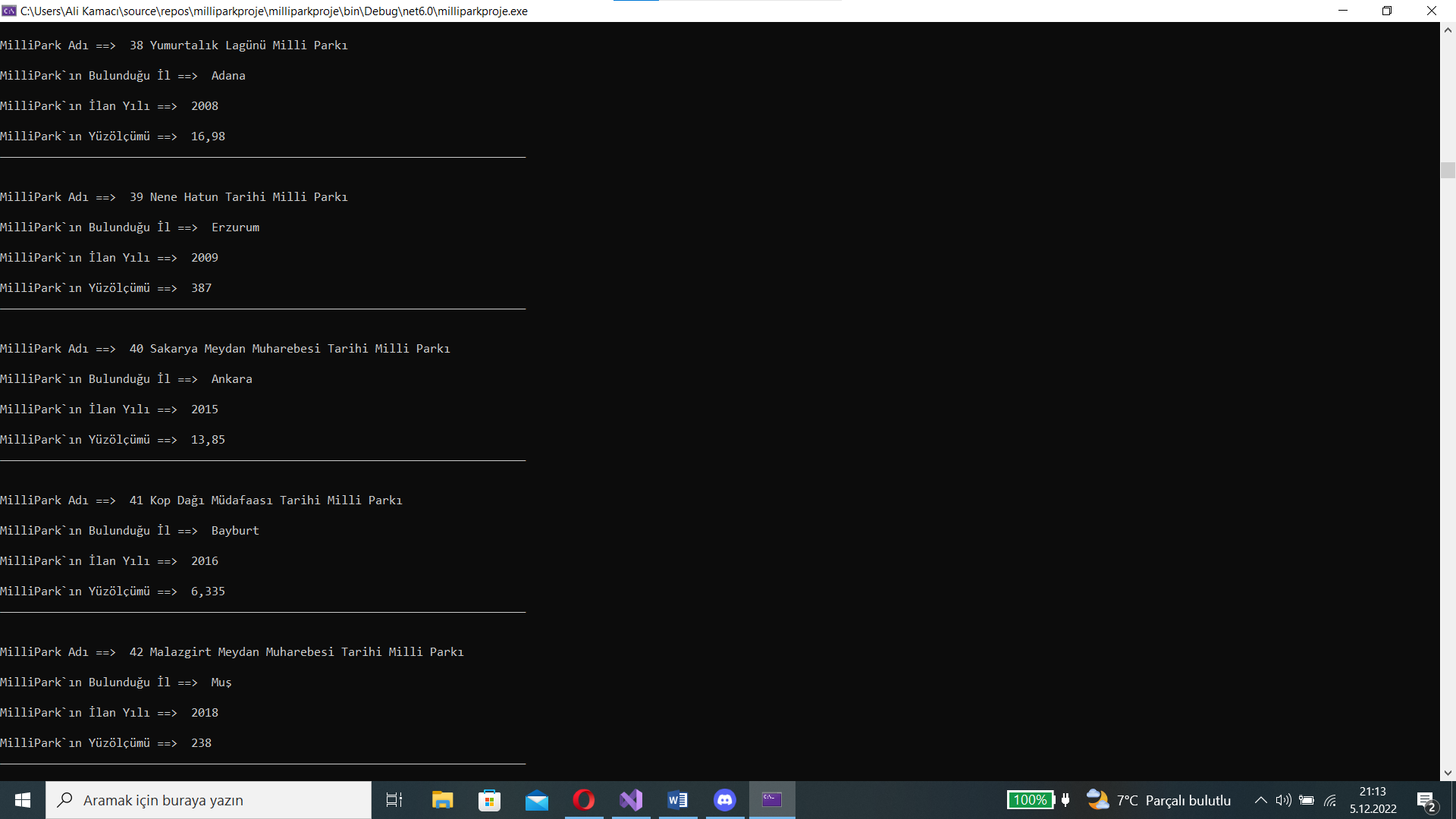
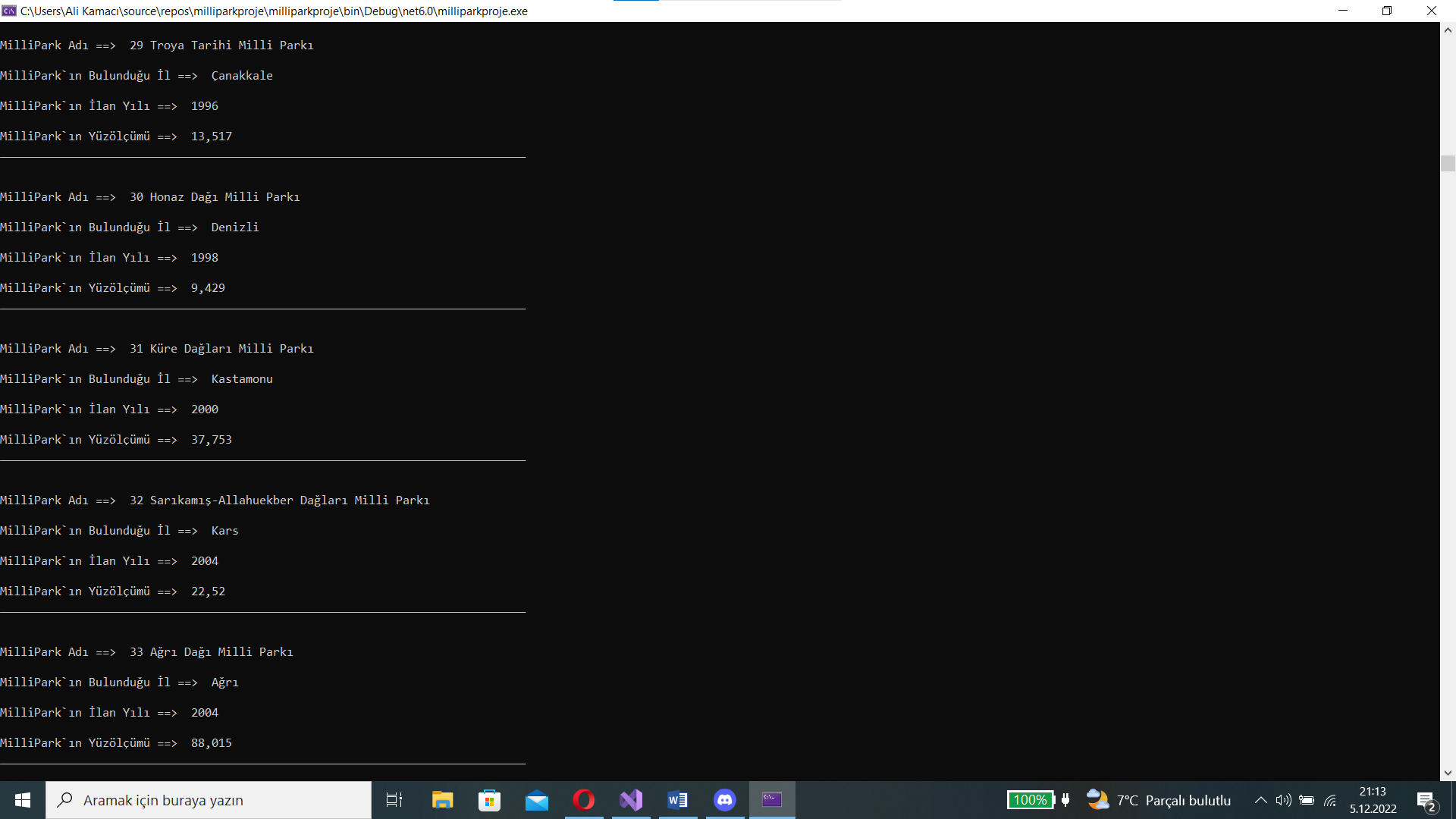
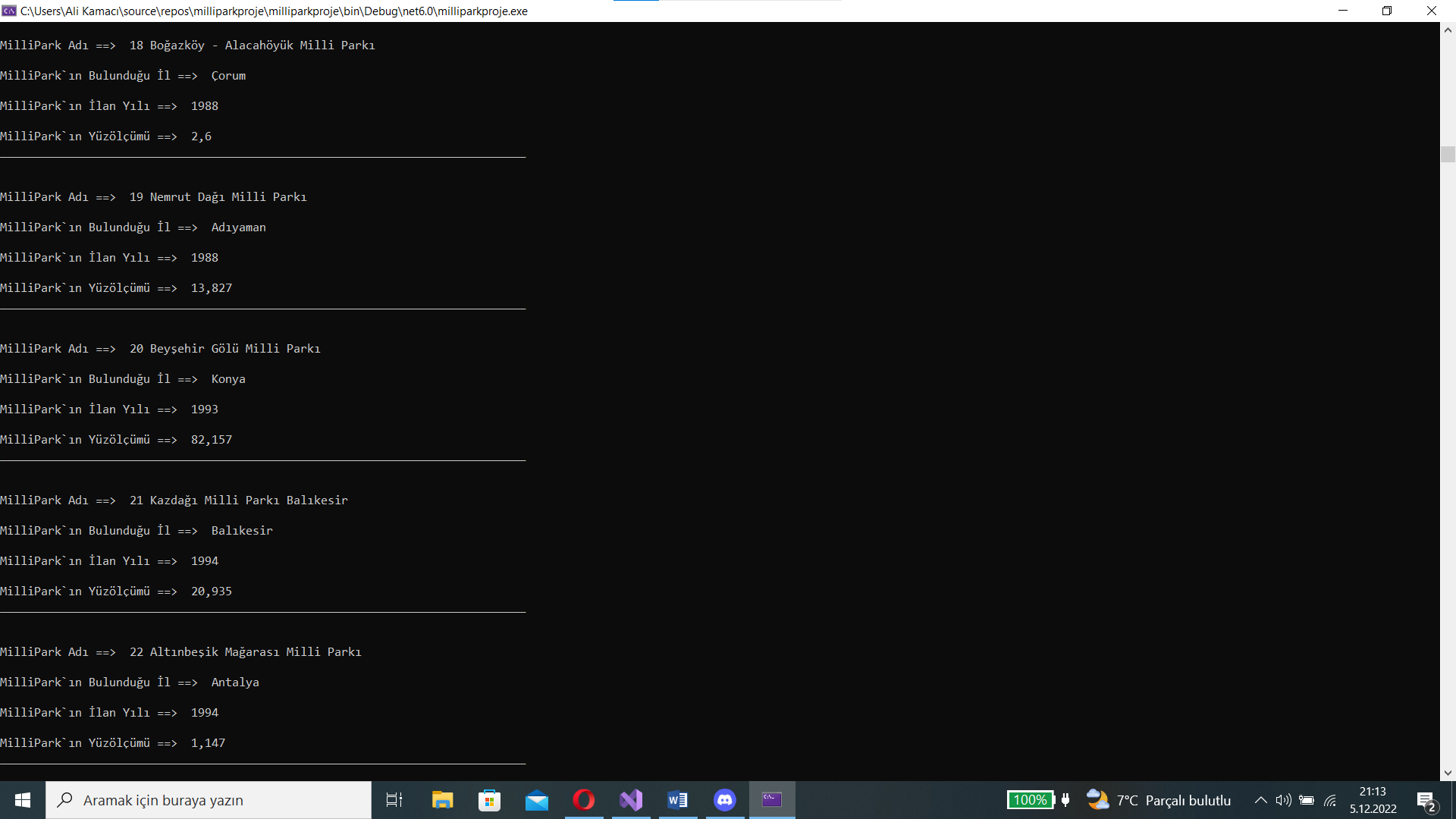
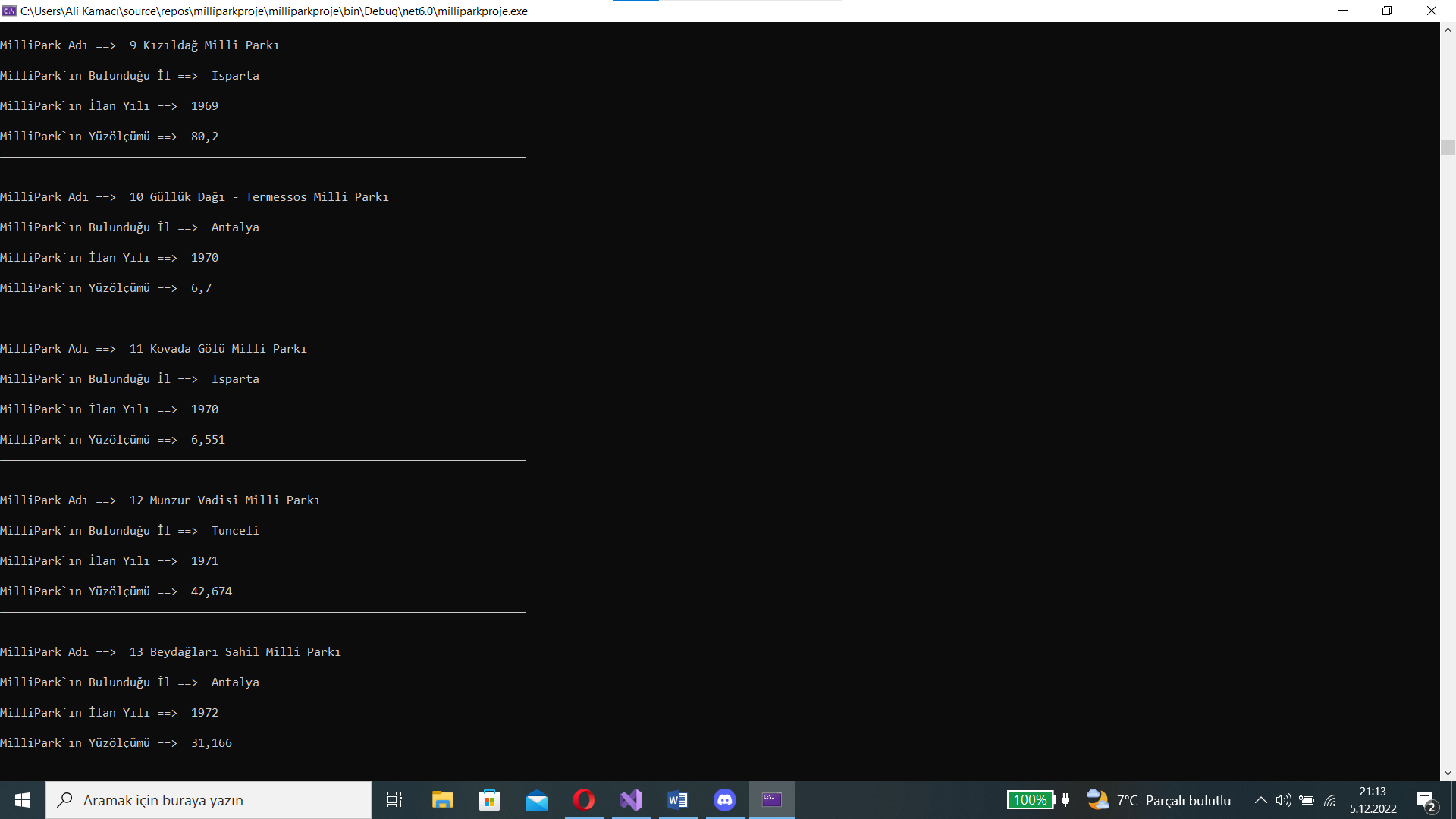
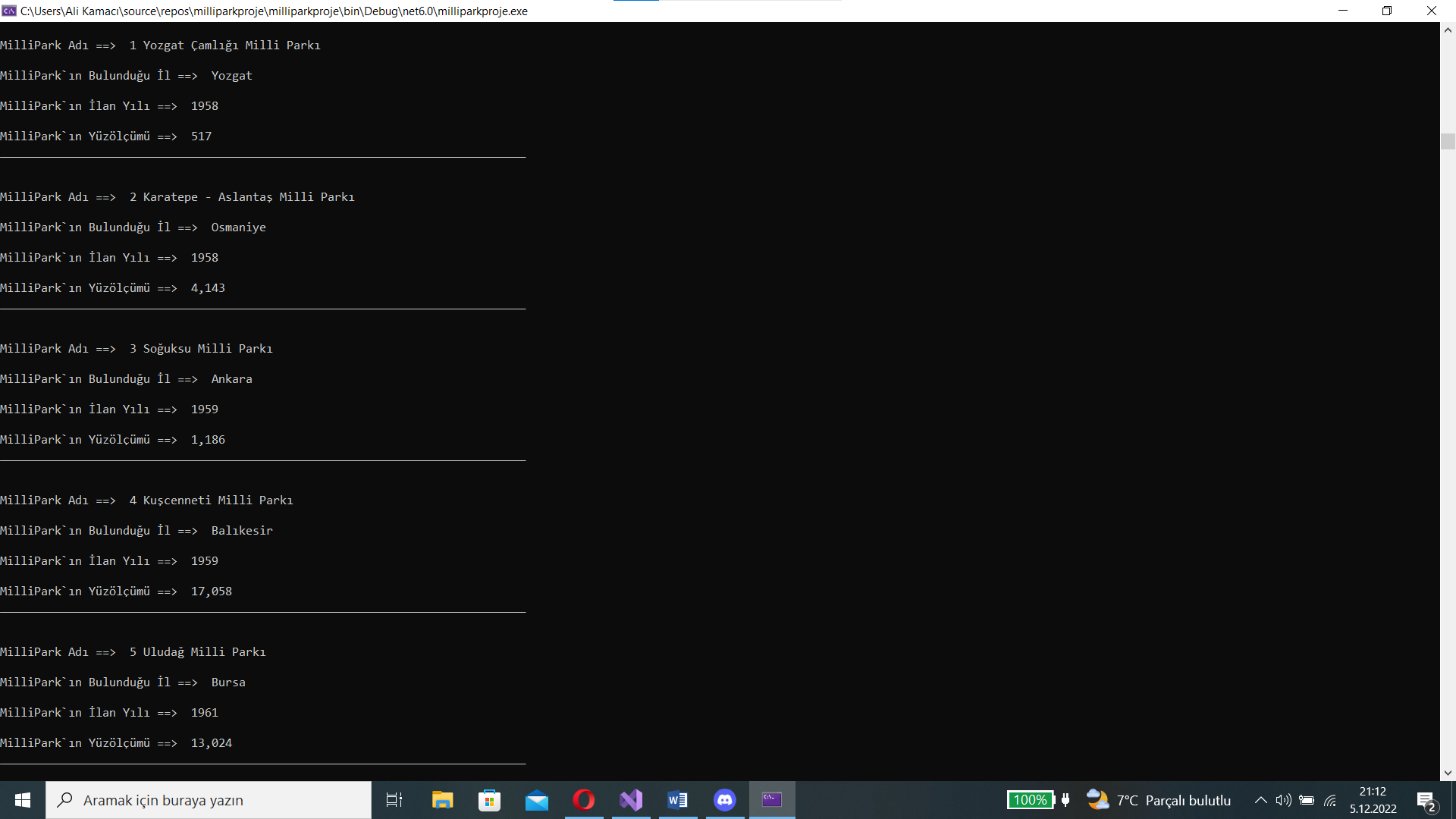
if (rear == maxSize - 1) // deal with wraparound

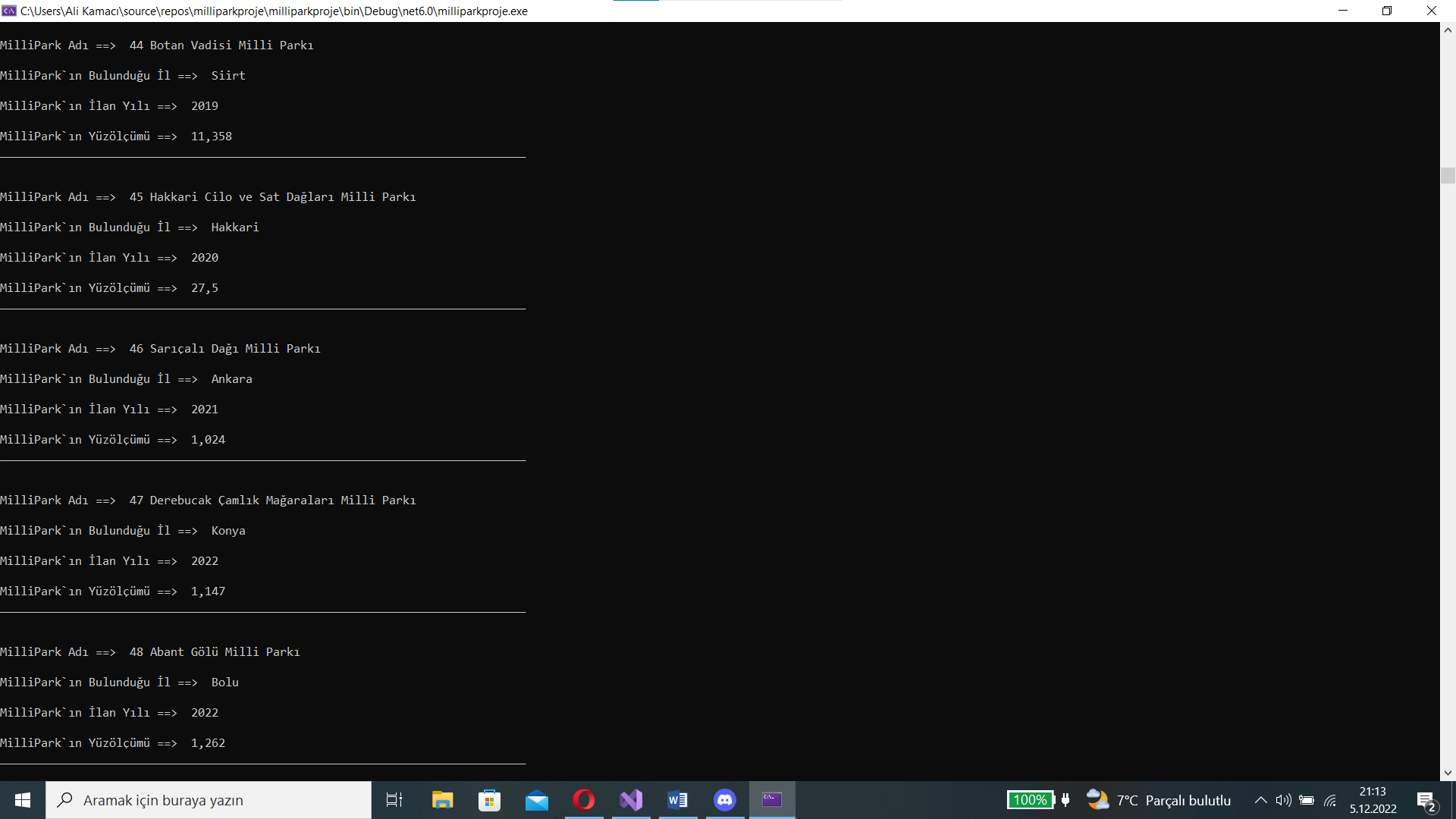
rear = -1;

queArray[++rear] = j; // increment rear and insert

nItems++; // one more item

### }2.b.2 Ekran Görüntüleri





# 3) Öncelikli Kuyruk

## 3.a Öncelikli Kuyruk (sınıfı) Oluşturma

### 3.a.1 Kaynak Kod

List<milliPark> liste;

public PQ() { liste = new List<milliPark>(); } //Constructor

public void PQEkle(milliPark milliParklar) //Listeye millipark elemanları eklendi.

{

liste.Add(milliParklar);

}

public void PQSil() //Liste küçükten büyüğe sıralandı ve silinip konsola bastırıldı.

{

List<milliPark> bastirListe = new List<milliPark>();

milliPark tempMP = liste.ElementAt(0); //Genel temp kullanımı için listenin başına atandı.

for (int i = 0; i < liste.Count; i++) //2 for döngüsü ile elemanların yüzölçümlerine göre sıralanması tamamlandı.

{

for (int j = i + 1; j < liste.Count; j++)

{

if (liste[i].Park\_yuzolcumu > liste[j].Park\_yuzolcumu) //İterator gezinimi yapıldı.

{ tempMP = liste.ElementAt(i);

liste[i] = liste[j];

liste[j] = tempMP;

}

}

}

while(!BosMu())

{

int i = 0;

Yazdir(liste[i]);

liste.RemoveAt(i);

i++;

}

}

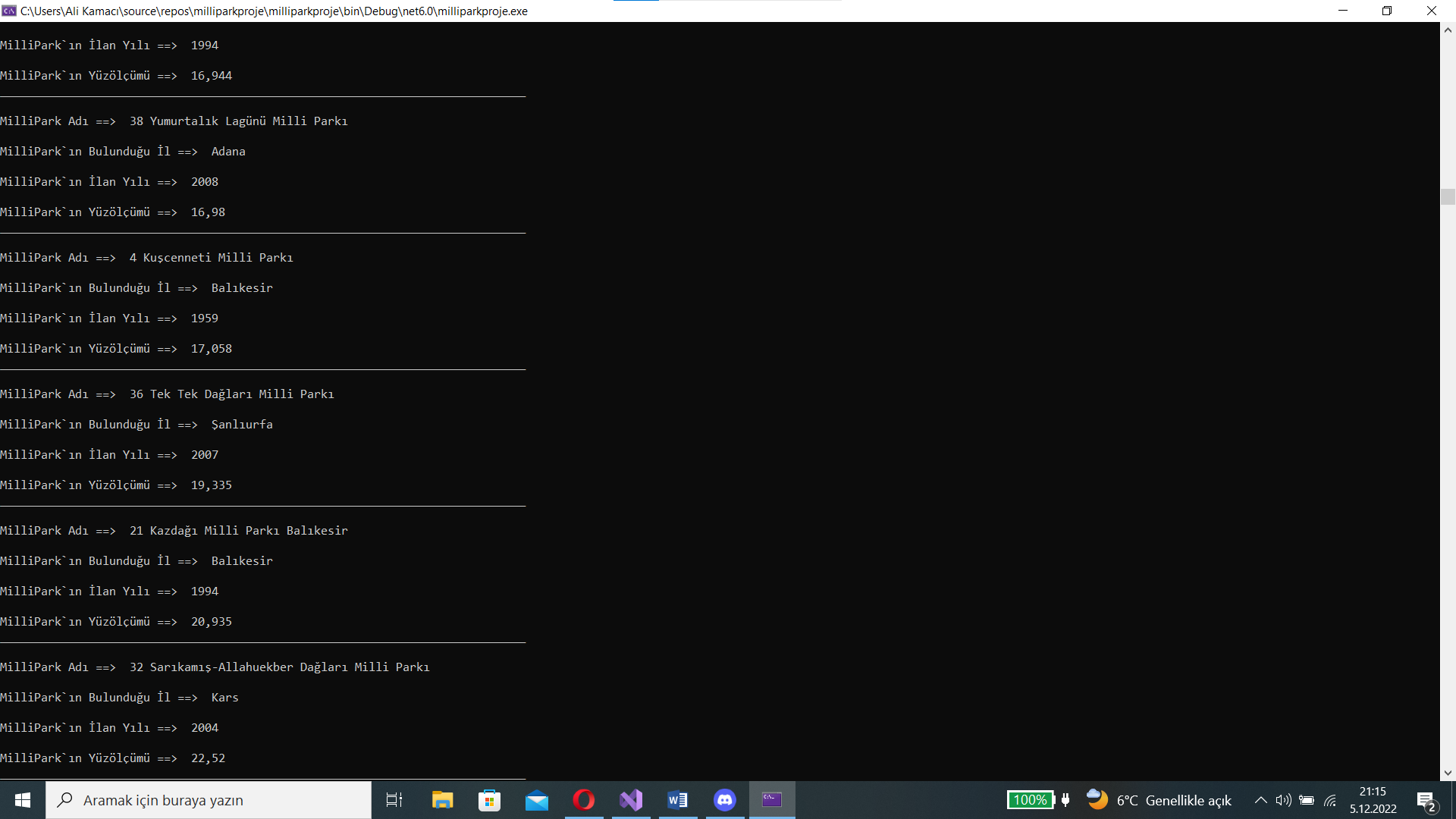
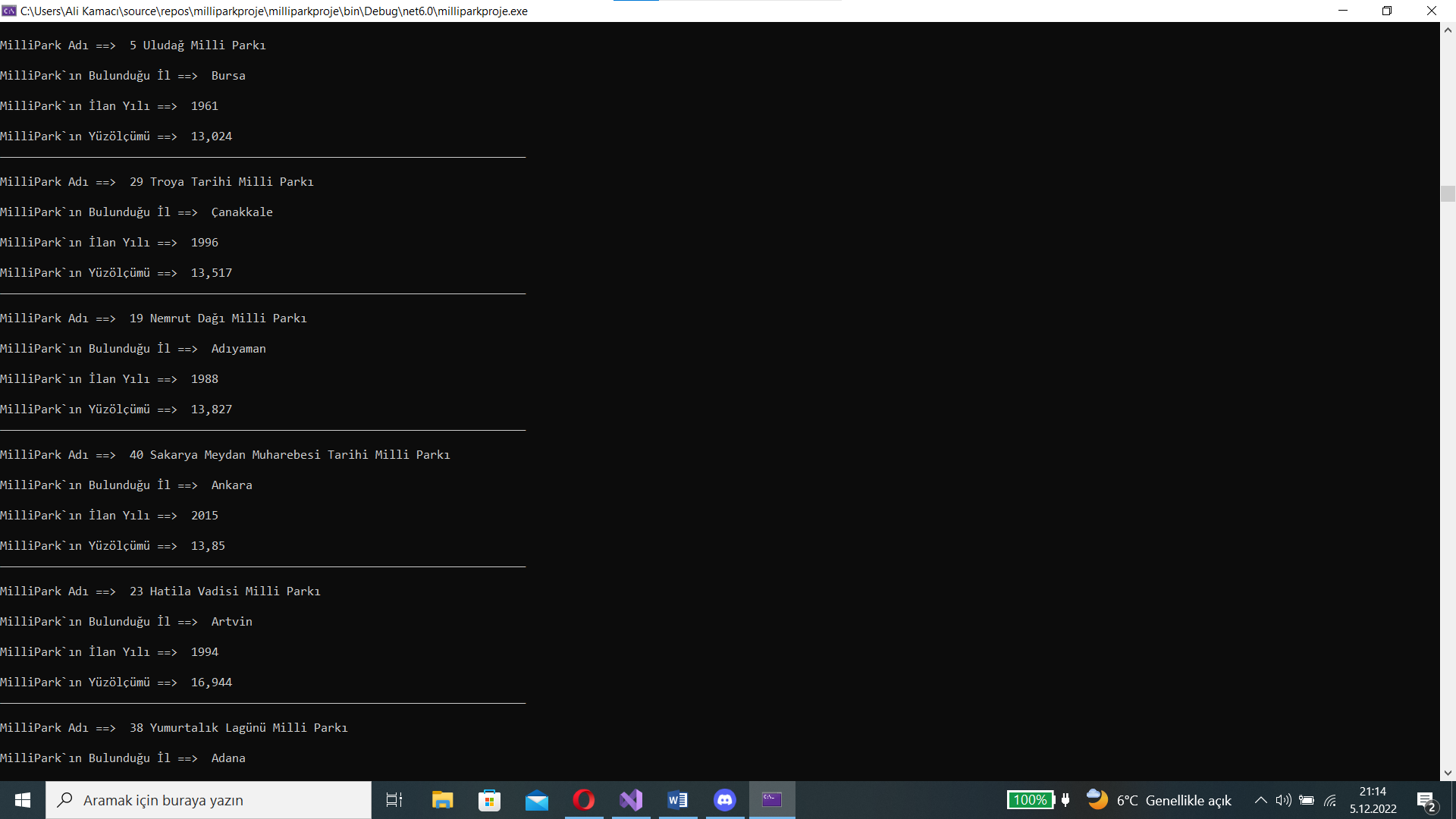
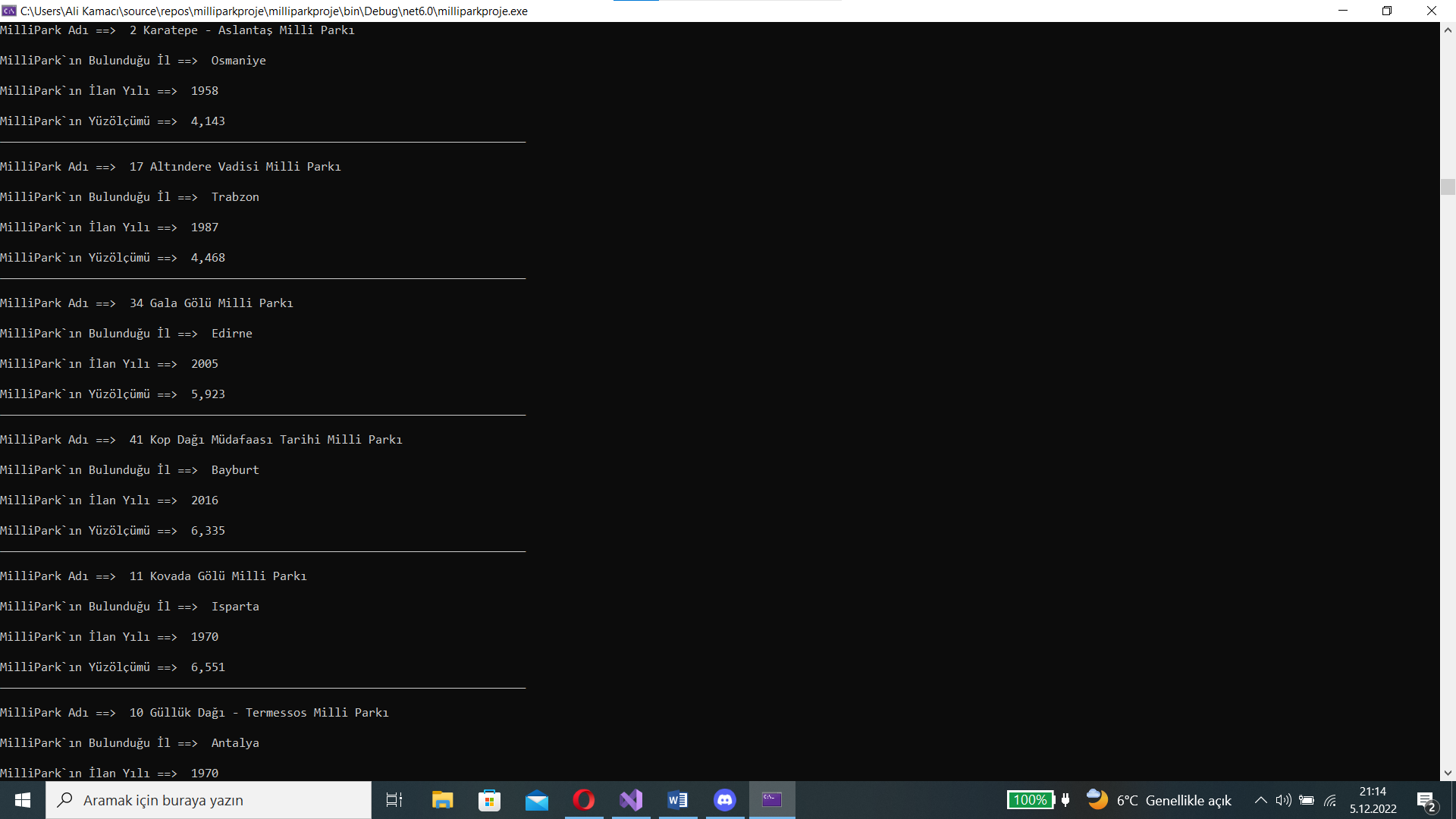
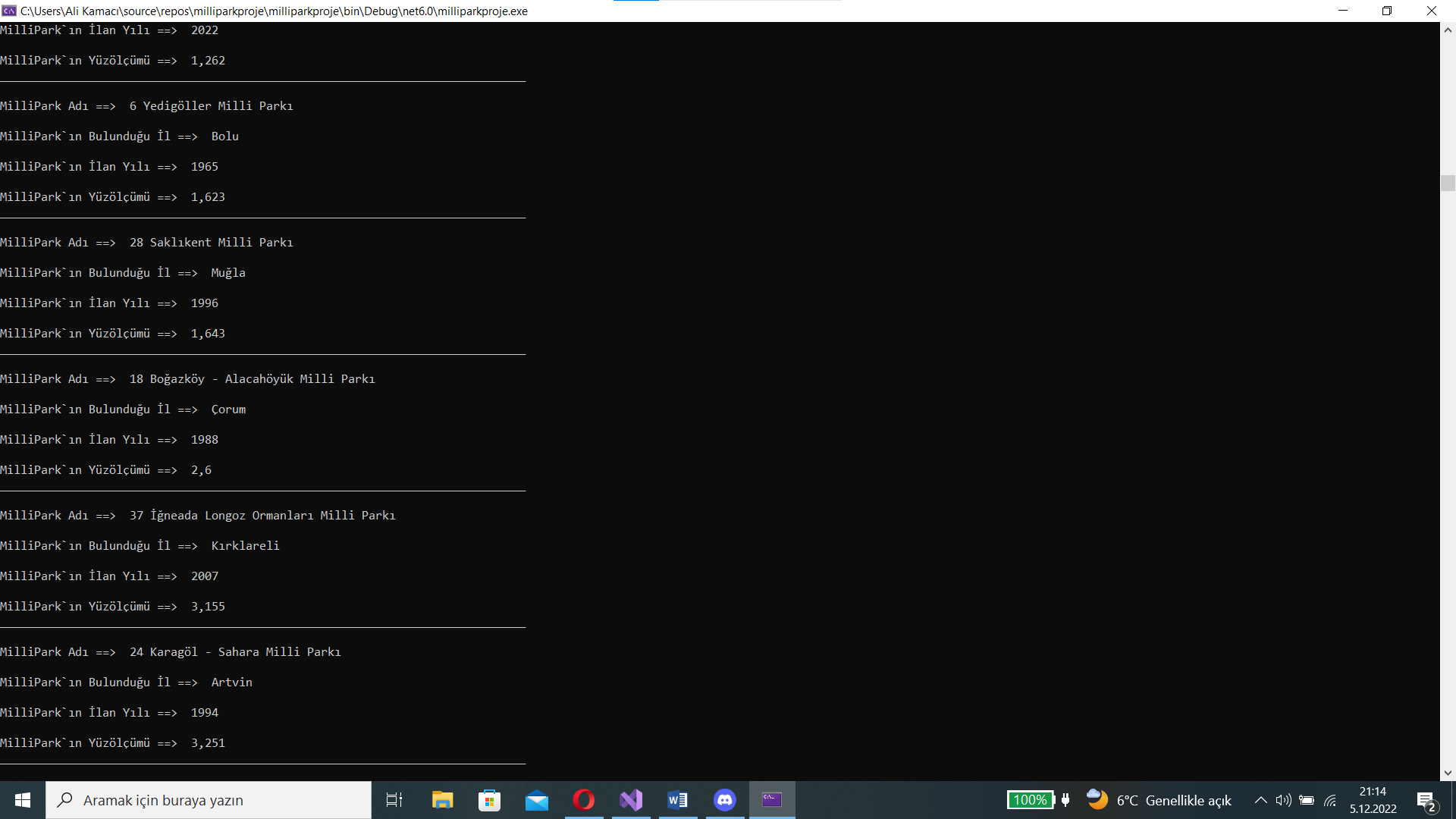
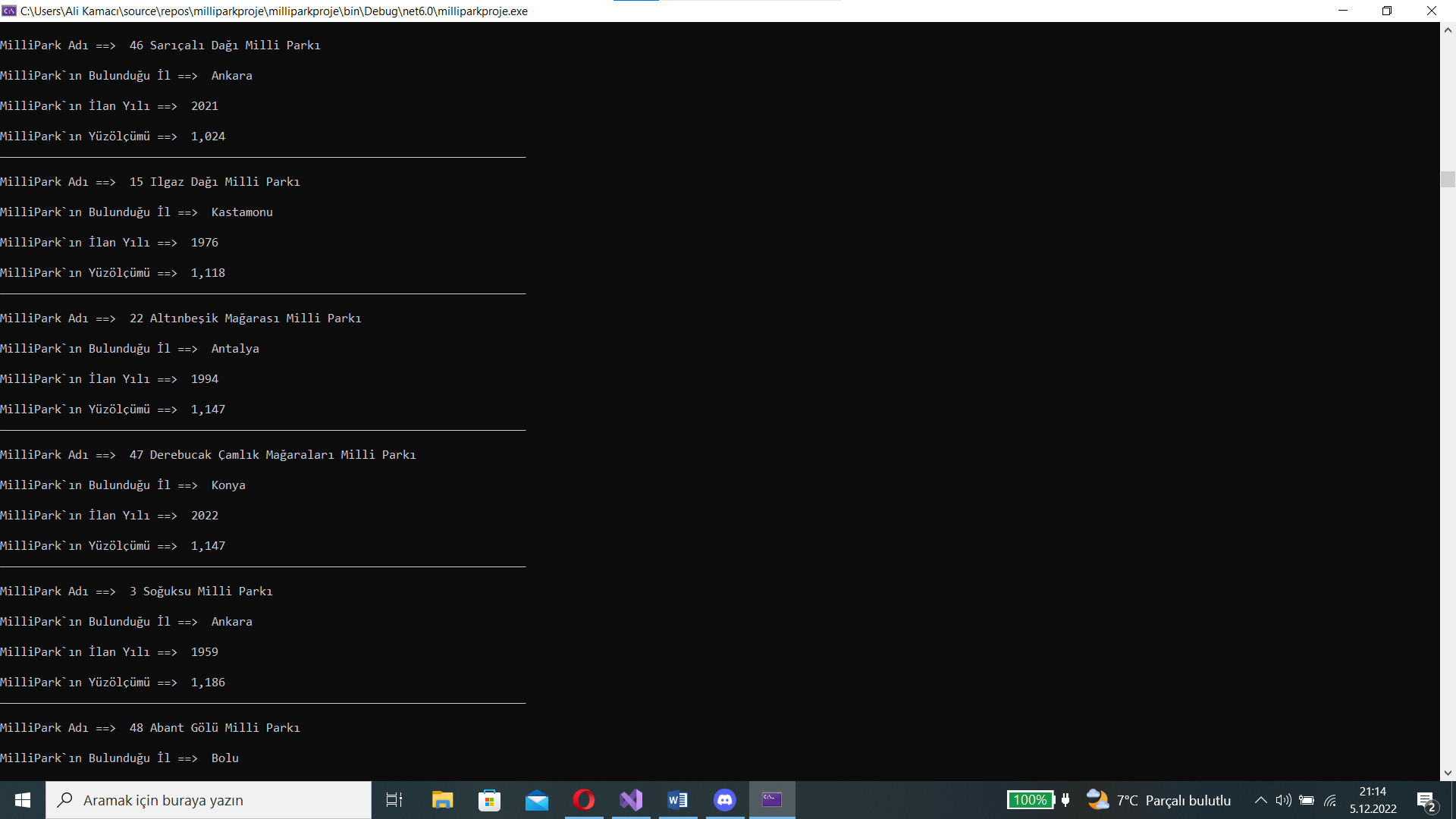
public bool BosMu() //Listenin boş olup olmadığını döndüren fonksiyon.

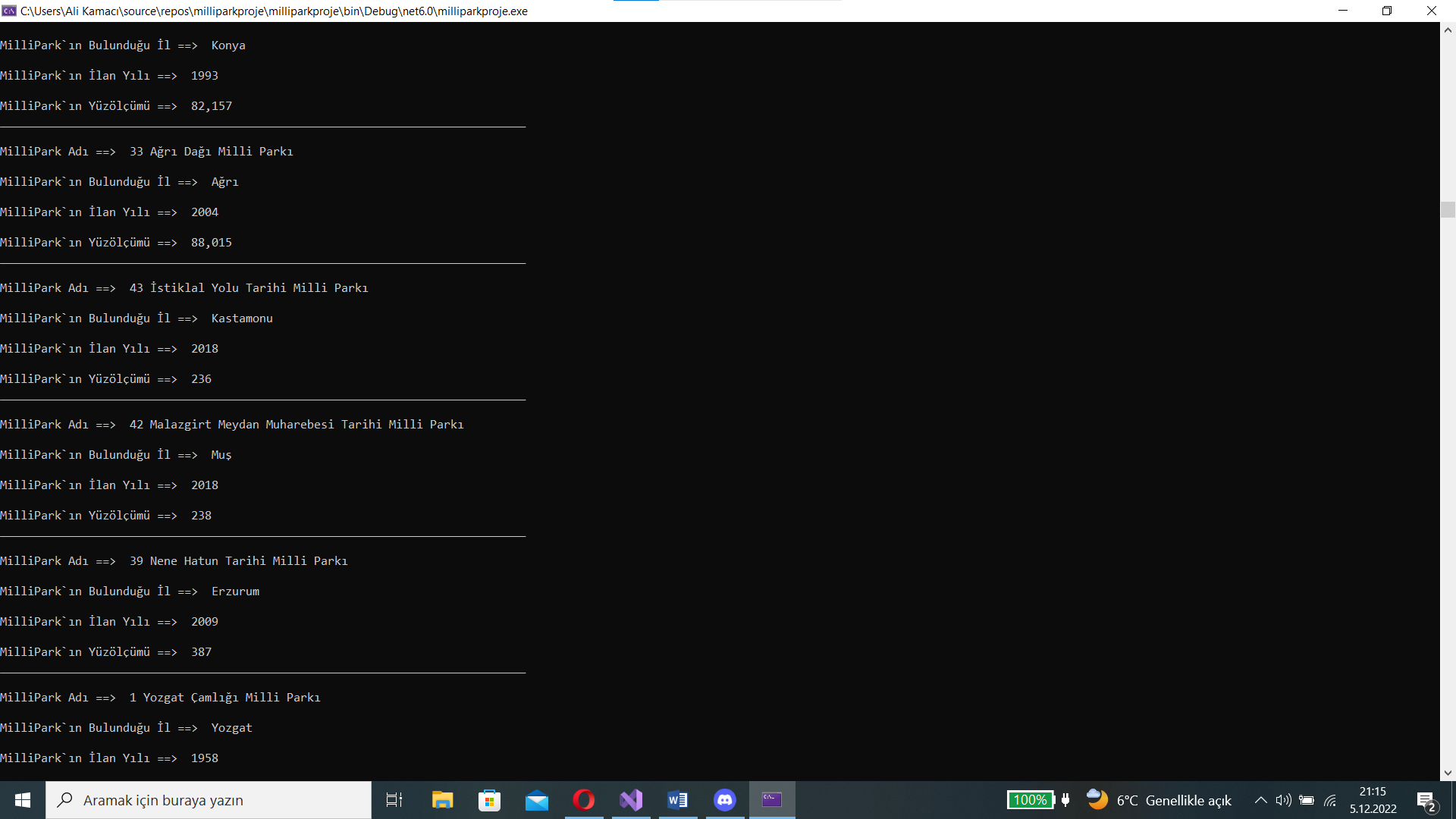
{

return liste.Count == 0;

}

### 3.a.2 Ekran görüntüleri





# 4) Kuyruk ve Öncelikli Kuyruk

## 4.a Kuyruk kullanarak süre hesaplanması

### 4.a.1 Kaynak Kod

private int maxSize; //queArray in maksimum alabileceği boyut

private int[] urunList; //Millipark nesnelerini kuyruk için tutan array

private int rear; //Arrayin sonunu tutan değişken

private int nItems; //Arrayin item sayısını tutan değişken

int islem\_sure;

int[] sureList = new int[11];

int[] pqsureList = new int[11];

public KasaKuyruk(int s) // constructor

{

maxSize = s;

urunList = new int[maxSize];

nItems = 0;

rear = 0;

}

public void insert(int j) // put item at rear of queue

{

islem\_sure = j \* 3;

if (rear == maxSize - 1) // deal with wraparound

rear = -1;

urunList[++rear] = islem\_sure; // increment rear and insert

Console.WriteLine("{0}.Müşterinin işlem süresi = {1}",nItems+1,islem\_sure);

nItems++; // one more item

sureList[0] = 18;

for (int i = 1; i < 11; i++) //Listeleri toplama işlemine soktuk.

sureList[i] = sureList[i - 1] + urunList[i];

if (nItems == 11)

{

int toplam\_sure = 0;

int kuyruk\_sure = 0;

foreach (int item in urunList)

{

toplam\_sure = toplam\_sure + item; // Kuyrukta geçen zaman

}

foreach (int sure in sureList)

{

kuyruk\_sure = kuyruk\_sure + sure; // Devinirli zaman

}

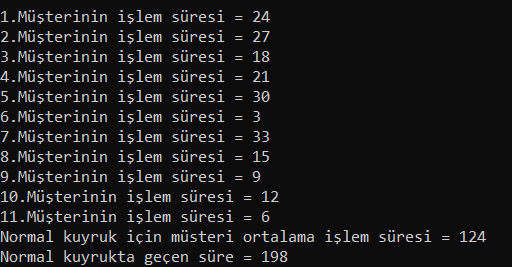
double ortalama\_sure = (kuyruk\_sure/11);

Console.WriteLine("Normal kuyruk için müsteri ortalama işlem süresi = {0}",ortalama\_sure);

Console.WriteLine("Normal kuyrukta geçen süre = {0}", toplam\_sure);

### }

### 4.a.2 Ekran görüntüleri



## 4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması

### 4.b.1 Kaynak Kod

public void oncelikli() //İşlem süresine göre küçükten büyüğe sıralı öncelikli kuyruk

{

int temp = urunList.ElementAt(0);

//Küçükten büyüğe sıralama yapıldı.

for (int i = 0; i < urunList.Length; i++)

{

for (int j = i + 1; j < urunList.Length; j++)

{

if (urunList[i] > urunList[j])

{

temp = urunList.ElementAt(i);

urunList[i] = urunList[j];

urunList[j] = temp;

}

}

}

foreach (int item in urunList)

{

Console.WriteLine("Sıralı Müşterinin işlem süresi = {0}",item);

}

pqsureList[0] = 3;

for (int i = 1; i < 11; i++)

pqsureList[i] = pqsureList[i - 1] + urunList[i]; //Üst kısımla tamamen aynı yapı.

if (pqsureList.Count() == 11)

{

int toplam\_sure = 0;

int kuyruk\_sure = 0;

foreach (int item in urunList)

{

toplam\_sure = toplam\_sure + item;

}

foreach (int sure in pqsureList)

{

kuyruk\_sure = kuyruk\_sure + sure;

}

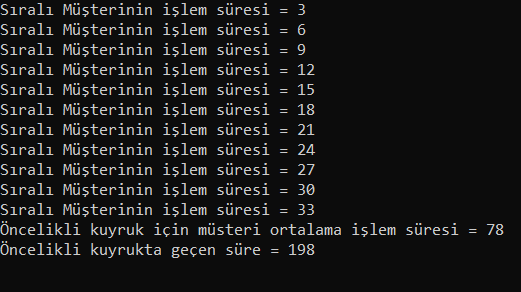
double ortalama\_sure = (kuyruk\_sure/11);

Console.WriteLine("Öncelikli kuyruk için müsteri ortalama işlem süresi = {0}", ortalama\_sure);

Console.WriteLine("Öncelikli kuyrukta geçen süre = {0}", toplam\_sure);

}

### 4.b.2 Ekran görüntüleri



## 4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması

Öncelikli kuyruk çok daha hızlı bir ortalama işlem süresine sahip.

Normal kuyrukta baştaki müşterilerin ürünlerinin sayısı az ise işlemleri uzun sürecek olan müşterileri bekleyeceğinden işlem tamamlanma süresi artar. Öncelikli kuyruk bu konuda çok daha iyidir.Bu durumun tersi ise öncelikli kuyruğun dezavantajıdır.

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) Sınıf Oluşturma** | **5** | 5 | Parametreli constructor oluşturuldu. |
| **1 b) Bileşik veri yapısının oluşturulması ve elemanların eklenmesi; Kaynak kod, ekran görüntüsü.** | **15** | 15 | Nesneler için özellikler listelerden çekildi.Get metodları ile erişildi ve nesne oluşturuldu. |
| **1 c) Veri yapısının elemanlarının listelenmesi ve 2 adet yüzölçümü değeri bulunması; Kaynak kodlar, ekran görüntüsü.** | **10** | 8 | Listelere yüzölçümlerine göre atamalar yapılıp adımlarda bastırıldı. En son tüm millipark nesneleri ve yüzölçüm toplmaları bastırıldı. |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **10** | 10 | Parkları yığıtlara koyduk.Yığıttaki parkları LIFO yapısı ile ekrana yazdırdık |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **10** | 8 | Parkları kuyruklara koyduk.Kuyruktaki parkları FIFO yapısı ile ekrana yazdırdık |
| **3 ) Öncelikli Kuyruk kod ve ekran görüntüleri** | **20** | 10 | Öncelikli metot oluşturuldu.  Artan sıradan tekrar süreler hesaplandı. |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri, karşılaştırma ve soruların cevapları.** | **20** | 20 | Öncelikli ve normal kuyruğa müşteri eklemeleri yapıldı.İşlem ,ort ve toplam süreler hesaplandı.Öncelikli kuyruğun önemi anlaşıldı. |
| **5) Özdeğerlendirme Tablosu** | **10** | 10 | İyi emek verdiğimi düşünüyorum. |
| **Toplam** | **100** | 96 |  |

**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.**