**1-1)**

Programın amacı, 2 kurala bağlı kalarak bir satranç tahtasını kaç farklı şekilde doldurabiliriz görmek.

**Kural-1:** Siyah Piyonla Başlama.

**Kural-2:** 2 tane siyah piyon yan yana gelmesin.

**1-2)**

**Kullanılan Fonksiyonlar;**

UsHesapla(int sutun)

BinaryDondur(int sayi)

StringTersDondur(string veri)

SonucDondur(string veri,int uzunluk)

Hesapla(int satir,int sutun)

**1-3)**

Program başladığında karşımıza satır ve sütun bilgilerinin girilmesi istenen bir form ekranı geliyor satır ve sütun bilgisi girildikten sonra butona basıldığında program ilk olarak,

“UsHesapla” fonksiyonunu aktif ediyor. “UsHesapla” fonksiyonunun görevi, 10’luk sistem ile girilen sütun sayısının 2’lik sistemde kullanılabilir olmasını sağlamak örnek olarak 10’luk sistemde 2 girersek bunu 22 şekline çevirip bize geri dönüş veriyor bu sayede kurulacak döngünün başlangıç ve bitiş değerleri belirleniyor.

Bu işlem sonrasında çıkan değerler “BinaryDondur” fonksiyonuna gönderiliyor. Bu fonksiyonun görevi, Girilen sayıyı binary sayılarına çevirmek örnek olarak 4 sayısını 001 şekline getiriyor.

Bu işlem sonrasında binary sayıların doğru forma getirilmesi için “StringTersDondur” fonksiyonu çalıştırılarak girilen stringi ters çevirmek örnek olarak yukarıdan 001 verildiğinde 100 şekline getiriyor.

Girilen sayıyı binary formatında düzgün hale getirdikten sonra problemin çözümü için kullanılan algoritmanın çalıştığı fonksiyon “SonucDondur”. Bu fonksiyonun görevi gelen verinin kurallara uyup uymadığını kontrol etmek. Kurala uyuyorsa true uymuyorsa false değeri dönüyor.

“Hesapla” fonksiyonun görevi, yukarıdaki fonksiyonları sırayla çalıştırmak ve “SonucDondur” fonksiyonundan gelen sonuca göre ekrana doğru sonucu yazdırmak.

**1-4)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Ödev1

{

public partial class frmPiyon : Form

{

public frmPiyon()

{

InitializeComponent();

}

public int UsHesapla(int sutun) //10'luk sistemden gelen sayıyı 2'lik sisteme getirilecek hale sokuyoruz

{

int toplam = 1;

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

{

toplam \*= 2; //2 üzeri girilen sayı şekline gelecek

}

return toplam;

}

public string BinaryDondur(int sayi) //Girilen sayıyı Binary biçimine çeviriyoruz

{

int sonuc;

string BinarySonuc = "";

while (true)

{

sonuc = sayi % 2; //girilen sayının modunu bir değişkene aktarıyoruz

sayi = sayi / 2; //sayıyı 2 ye tam bölerek güncelliyoruz

BinarySonuc += sonuc.ToString(); //değişkene aktarılan rakamı string olarak bir değişkene aktarıyoruz

if (sayi < 2) //sayı 2'den küçük ise artık bölmenin anlamı yok

{

BinarySonuc += sayi.ToString(); //kalan sayıyıda string değişkene aktarıyoruz

break; //sayı ile işimiz bittiği için while döngüsünden çıkıyoruz

}

}

BinarySonuc = StringTersDondur(BinarySonuc); //string değişkeni ters döndürerek binary formuna ulaşmış oluyoruz

return BinarySonuc; //sonucu döndürüyoruz

}

public string StringTersDondur(string veri) //Verilen string ifadeyi ters çeviriyoruz (kütüphanede olan fonksiyon var ama biz kendimiz yazmak istedik)

{

int uzunluk = veri.Length; //girilen stringin uzunluğunu değişkene aktarıyoruz

string yedek = "";

for (int i = 0; i < uzunluk; i++) //stringin uzunluğu kadar döngüyü döndürüyoruz

{

yedek += veri[uzunluk - (i + 1)]; //ters çevirme işlemi

}

return yedek;

}

public bool SonucDondur(string veri, int uzunluk) //2 tane 0 yan yana var mı kontrolu yapılıyor

{

bool sonuc = false;

for (int i = 0; i < uzunluk - 1; i++)

{

if ((veri[i] == '0') && (veri[i + 1] == '0')) //eğer 2 tane yan yana sıfır var ise false yok ise true değeri döndürüyoruz

{

sonuc = false;

break;

}

else

{

sonuc = true;

}

}

return sonuc;

}

public void Hesapla(int satir,int sutun) //satir ve sütun bilgilerine göre işlemi yapan ana fonksiyon

{

int dongubaslangic, dongubitis, uzunluk, sayac = 0;

dongubitis=UsHesapla(sutun); //döngünün bitiş değerini buluyoruz

dongubaslangic = dongubitis / 2; //döngünün başlangıç değerini bitiş değerinin yarısından alıyoruz

//bunu yapmamızın nedeni binary formatından olan verilerin başlarında hep 1 olmasını istiyoruz

//örnek 32=100000 , 16=10000

string binarykarsilik = "", mesajsonucu = "";

for (int i = dongubaslangic; i < dongubitis; i++)

{

binarykarsilik = BinaryDondur(i); //döngünün i.cisini binary'ye çeviriyoruz değişkene aktarıyoruz

uzunluk = binarykarsilik.Length; //çıkan stringin uzunluğunuda başka bir değişkene aktarıyoruz

if (SonucDondur(binarykarsilik, uzunluk)) //sonuc dndur fonksiyonunu çağırıyoruz eğer sonuc true çıkarsa aradığımız değer bu demektir

{ //sayacı arttırıp ekrana toplu yazmak için bir değişkene aktarıyoruz

sayac++;

mesajsonucu += binarykarsilik + "\n";

}

}

if (satir>1) //eğer satır sayısı 1'den büyük ise ilk önce 1 satırda çıkan sonucu yazdırıp sonra girilen satır bilgisine göre çarpıp sonucu yazdırıyoruz

{

MessageBox.Show(satir + "x" + sutun + " Olasılıkları ;\n\n" + mesajsonucu + "\nTek Satırlık Olasılık : " + (sayac) + "\nToplam Olasılık : " + (sayac \* satir));

}

else //satır sayısı 1 den küçük ise sadece sonucu yazdırıyoruz

{

MessageBox.Show(satir + "x" + sutun + " Olasılıkları ;\n\n" + mesajsonucu + "\nToplam Olasılık : " + (sayac \* satir));

}

}

private void btnHesapla\_Click(object sender, EventArgs e)

{

try //hataları azaltmak için try catch kullanıyoruz

{

int satir = Convert.ToInt32(txtsatir.Text);

int sutun = Convert.ToInt32(txtsutun.Text);

if (sutun==1) //sütun bilgisi 1'e eşit ise verilecek mesaj

{

MessageBox.Show("Tek Piyon İle İşlem Yaparsak Sadece Beyaz Piyon Yerleştirilir.");

}

else if ((satir<0)||(sutun<0)) //eksik giriş var demektir

{

MessageBox.Show("Girişleri Doldurduğunuzdan Emin Olunuz..");

}

else //sıkıntı yok ise fonksiyonu çalıştır

{

Hesapla(satir, sutun);

}

}

catch (Exception) //hata çıkarsa verilecek mesaj

{

MessageBox.Show("Girişleri Sayılar İle Doldurduğunuzdan Emin Olunuz..");

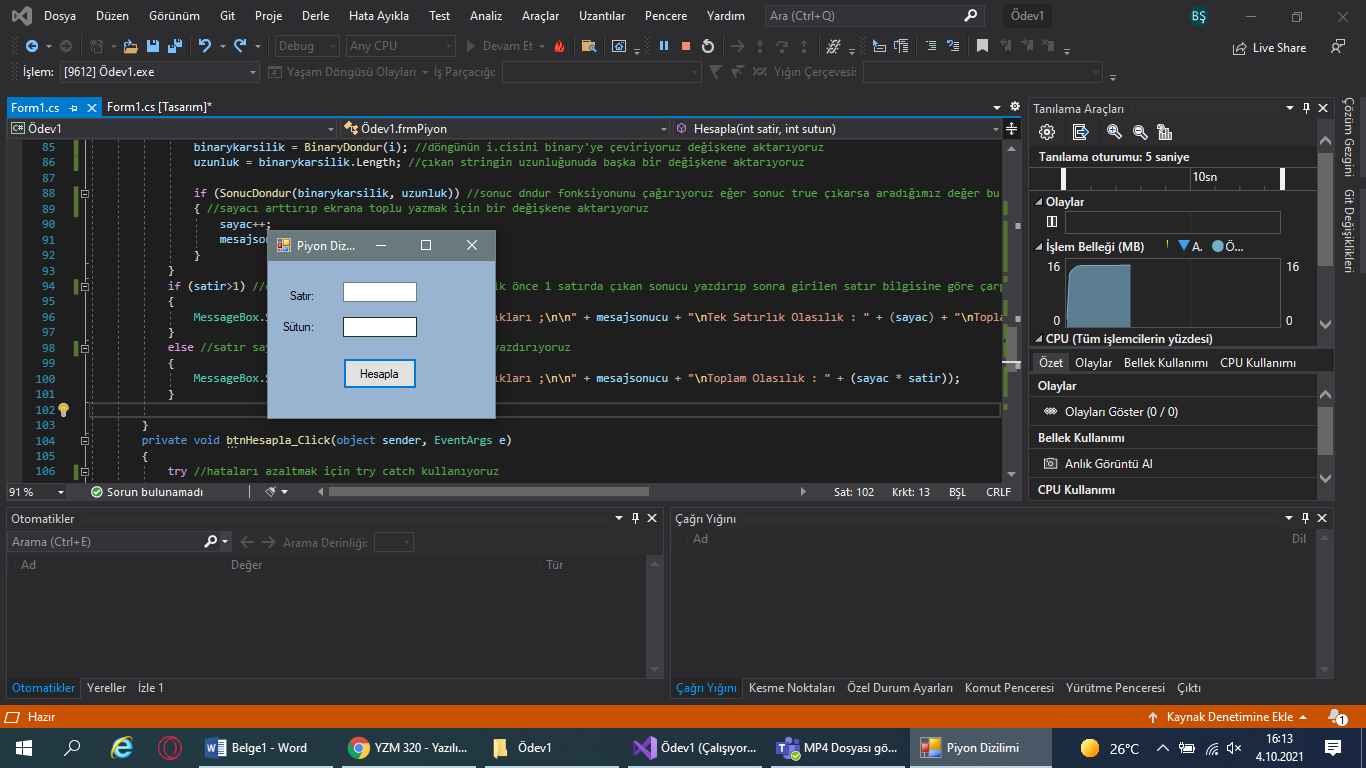
}

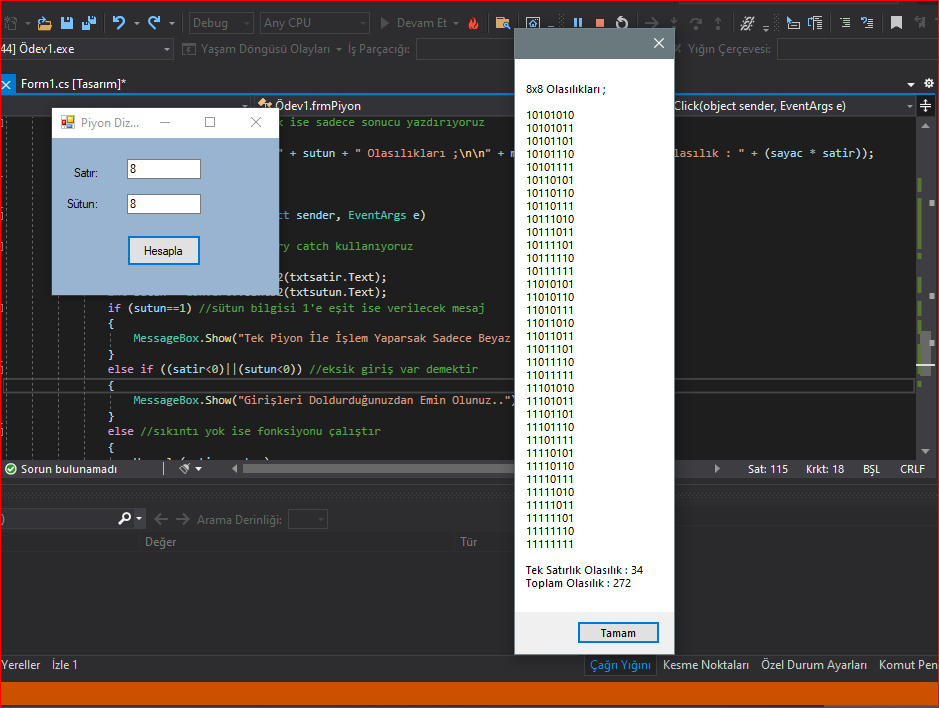
}

}

}

**1-5)**





**1-6)**

|  |  |
| --- | --- |
| **Risk Level** | Yüksek |
| **Purpose** | Gerçekleştirilen programın doğru bir şekilde  çalıştığının gösterilmesi |
| **Inputs** | 8x8 |
| **Expected**  **Outputs** | 272 |
| **Pass Criterias** | 272 |
| **Fail Criterias** | 272 dışındaki bir sayının program tarafından  oluşturulması. |
| **Test Procedure** | 1. Programı çalıştır 2. Tahtanın boyutlarını belirle 3. Program tarafından gelen sonuca bak 4. Sonucu beklenen sonuç ile karşılaştır 5. Eğer beklenen sonuç ile çıkan sonuç aynı ise test başarılı olarak isimlendir |