ZEYNEL BAYHAN 23011104 | GRUP 2

PROBLEMİN TANIMI VE ALGORİTMA

Problemin Tanımı

Kullanıcıdan alınan değişken bir değerden oluşturulan tahtaya; 3 oyuncunun Kırmızı, Sarı ve Mavi taşları yerleştirilerek oynanan, tahtanın son durumunda en çok taşı olan kullanıcının kazandığı bir programın yazılması.

Algoritma

1. **Kullanıcıdan tahtanın boyutu alınır.**
2. **İlk taşın ortadaki haneye koyulması sağlanır**
3. **Konan taşın diğer taşlara bitişik ve tahtada olma şartı aranır**
4. **Konan taş ile aynı renkte kalan ilk taş arasında kalan renkler konan taşın renklerine çevrilir.**
5. **Renklere göre taş sayıları ve en çok taşa sahip oyuncu kazanan olarak belirlenir.**
6. **Son durum ekrana yazdırılır ve oyun biter.**

Video linkim: https://youtu.be/hmXRlnmONuE

RAPORUM

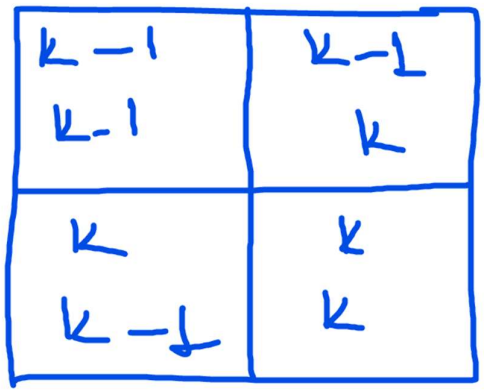
```c
#include <stdio.h>
#define SIZE 23
int main(void)
{
    int boyut, i, j, N, K, m, n, k, x, y, temp_x, temp_y, kirmizi, sari, mavi, kazanan, flag;
    char A[SIZE][SIZE];

    printf("Ilk olarak lutfen oyun tahtasinin boyutunu giriniz.\n");
    scanf("%d", &boyut);
    while(boyut > 23){
        printf("Oyun tahtasinin boyutu 23'den buyuk olamaz lutfen tekrar giriniz.\n");
        scanf("%d", &boyut);
    }

    for (i = 0; i < boyut; i++)
    {
        for (j = 0; j < boyut; j++)
        {
            A[i][j] = '-';
        }
    }
```

Kodumun Precessor kısmında ilk olarak stdio.h kütüphanemi dahil ettim ve tahtamın boyutunu en fazla 23 olarak belirttim (taşma olmaması için). Daha sonra değişkenlerimi ve karakter dizimi tanımladım ve kullanıcıdan oyun tahtasının boyutunu girmesini istedim. Eğer girdiği boyut 23'ten büyükse istenilen sayı girilene kadar döngüye soktum ve dizimin boş yerlerini ifade etmek için ' - ' kullandım.

Daha sonra Kullanıcının ilk taşı tahtanın tam ortasına koyması gerekiyor fakat bunun için iki farklı durum var: boyutun tek veya çift olması.

```c
if (boyut % 2 == 0) // Eger girilen boyut degeri cift ise 4 haneye yerlestirebiliriz.
{
    k = boyut / 2;
    printf("1. Oyuncu lutfen 'Kirmizi' tas icin koordinat giriniz.\n");
    scanf("%d %d", &n, &m);
    while (!((n - 1 == k - 1 && (m - 1 == k - 1 || m - 1 == k)) || (n - 1 == k && (m - 1 == k - 1 || m - 1 == k))))
    {
        printf("1. oyuncu lutfen 'Kirmizi' tasi tahtanin ortasina koyunuz.\n");
        scanf("%d %d", &n, &m);
    }
    A[n - 1][m - 1] = 'K';
    for (i = 0; i < boyut; i++)
    {
        for (j = 0; j < boyut; j++)
        {
            printf("%c ", A[i][j]);
        }
        printf("\n");
    }
```

Burada boyutumun çift olduğu durumda girilen koordinat n ve m değerleri için;
Boyutu 2'ye bölerek bunu k değişkenine depoladım. k değişkeni boyut = 6 için 3 olacak ve ben bunu k-1 şeklinde ifade edip istenilen 4 kutudan sol üstteki olacak şekilde belirttim.



Daha sonra ise bu 4 kutudan herhangi birine karşılık gelmeyecek bir değer girilirse while döngüsüne soktum ve bu 4 kutunun dışını değil ifadesi (!) ile hallettim.

Boyutun tek olması durumunda ise girilecek tek bir kutu var o da n ve m için (n/2+1 , m/2+1).

```
else // Eger girilen boyut degeri tek ise ilk tas boyut degerinin ortancasina yerlestirilir. (ornek: 7+1/2=4)
{
    k = (boyut / 2);
    printf("1. Oyuncu lutfen 'Kirmizi' tas icin koordinat giriniz.\n");
    scanf("%d %d", &n, &m);
    while (!(n - 1 == k && m - 1 == k)) // n ve m girilen satır ve sutun indeksleri olmak uzere bunlardan 1 cikararak dizideki asil yerlerini belirttim.
    {
        printf("1. oyuncu lutfen 'Kirmizi' tasi tahtanin ortasina koyunuz.\n");
        scanf("%d %d", &n, &m);
    }
    A[n - 1][m - 1] = 'K';      // 1. Oyuncunun kullandigi tas Kirmizi.
    for (i = 0; i < boyut; i++) // 1. Oyuncunun yerlestirdigi tas sonrasi tahta duzeni.
    {
        for (j = 0; j < boyut; j++)
        {
            printf("%c ", A[i][j]);
        }
        printf("\n");
    }
}
// Girilen ilk degerin bitisi
```

Burada ise k 'yı boyutun yarısı olarak kullandım bu da boyut = 7 için 3'e tekabül ediyor (-1 çıkarmadım çünkü zaten matris 0'dan başladığı için istenilen yerde.). Ve girilen n ve m değerleri için 'n-1 = k && m-1 = k' olmadığı sürece kullanıcıdan doğru inputu girmesini istedim.

```
N = 2; // N adim sayisi, boyut*boyut ise toplam yapilacak hamle sayisi. (ornek: 7*7=49)
while (N <= boyut * boyut)
{
    x = N % 3;
    if (x == 1)
    {
        printf("1. Oyuncu lutfen 'Kirmizi' tasi yerlestirmek icin koordinat giriniz. \n");
    }
    else if (x == 2)
    {
        printf("2. Oyuncu lutfen 'Sari' tasi yerlestirmek icin koordinat giriniz. \n");
    }
    else
    {
        printf("3. Oyuncu lutfen 'Mavi' tasi yerlestirmek icin koordinat giriniz. \n");
    }
```

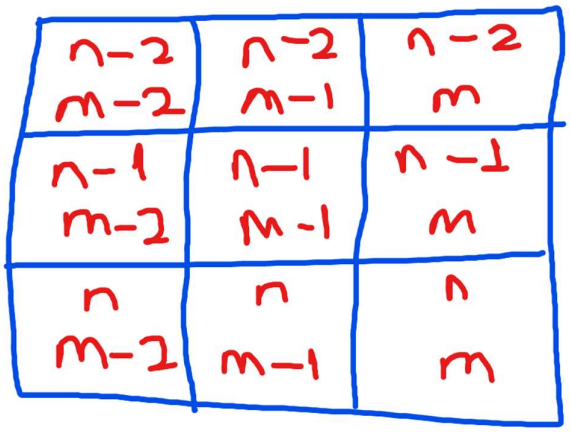
Burada ise asıl döngüyü başlattım. N değerini adım sayısı olarak belirledim ve mod operatörünü kullanarak hangi oyuncunun ne zaman hangi tası kullanacağını output olarak verdim. Adım sayısını ise boyut*boyut değeri olarak belirledim.

Daha sonra ise yerleştirmek istenen taşın tahtada uygun yere yerleştirildiğini kontrol ettim. Eğer n veya m değeri boyuttan büyük olursa tekrar koordinat isteyecek. Veya girilen değerdeki kutu boş değilse yani '-' değeri varsa tekrar koordinat isteyecek.

```
while(n > boyut || m > boyut){
    printf("Tasiniz tahtaya yerlesmedi, lutfen tekrar koordinat giriniz. \n");
    scanf("%d %d", &n, &m);
}

while (A[n - 1][m - 1] != '-')
{
    printf("Bu kutucuk zaten dolu, lutfen tekrar koordinat giriniz. \n");
    scanf("%d %d", &n, &m);
}
```

```
while (!(A[n - 2][m - 2] == 'K' || A[n - 2][m - 2] == 'S' || A[n - 2][m - 2] == 'M' ||
         A[n - 2][m - 1] == 'K' || A[n - 2][m - 1] == 'S' || A[n - 2][m - 1] == 'M' ||
         A[n - 2][m] == 'K' || A[n - 2][m] == 'S' || A[n - 2][m] == 'M' ||
         A[n - 1][m - 2] == 'K' || A[n - 1][m - 2] == 'S' || A[n - 1][m - 2] == 'M' ||
         A[n - 1][m] == 'K' || A[n - 1][m] == 'S' || A[n - 1][m] == 'M' ||
         A[n][m - 2] == 'K' || A[n][m - 2] == 'S' || A[n][m - 2] == 'M' ||
         A[n][m - 1] == 'K' || A[n][m - 1] == 'S' || A[n][m - 1] == 'M' ||
         A[n][m] == 'K' || A[n][m] == 'S' || A[n][m] == 'M')) // eger girilen (n-1,m-1) adresinin et
{
    printf("Bu kutucuk diger kutularla baglantili degil, lutfen tekrar koordinat giriniz. \n");
    scanf("%d %d", &n, &m);
}

if (x == 0) // tum kontrolleri yaptik ve dogru tas yerlestirdik.
{
    A[n - 1][m - 1] = 'M';
}
else if (x == 1)
{
    A[n - 1][m - 1] = 'K';
}
else
{
    A[n - 1][m - 1] = 'S';
}
```

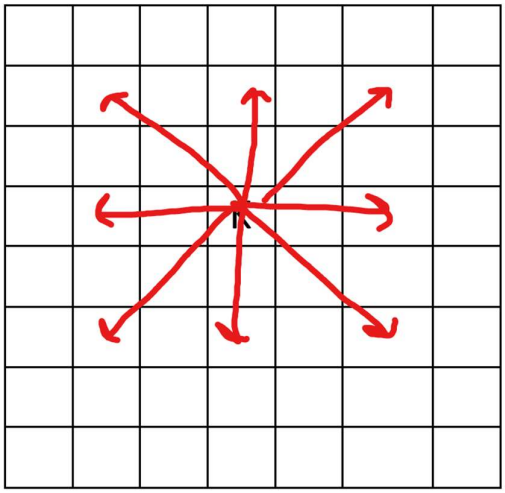
Bu kısımda ise eğer girilen input değerinin etrafındaki kutucuklarda K, M, S değerlerinden biri yok ise bağlantılı olmadığını anlıyoruz ve tekrar koordinat istiyoruz.



Bu şekilde girilen değeri kontrol ettirdim. n ve m değeri için matriste n-1,m-1 değerine karşılık gelecek.

Ve mod operatörü ile adım sayısını 1 ve 2 ye bölerek kalan sayısı eğer 1 ise Kırmızı, 2 ise Sarı ve kalansız ise Mavi taşı yerleştirdim.

Şimdi taşları doğru ve sorunsuz bir şekilde yerleştirdik. Sırada ise K M S taşlarının dönüşümü var.



K, M, S dönüşümünde ise girilen değerin sırasıyla sola, sol yukarıya, yukarıya, sağ yukarıya, sağa, sağ alta, aşağıya, sol aşağıya kontrolünü yaptırdım.

```
// sola dogru kontrol
flag = 0;
x = m - 3;
while (x >= 0 && flag == 0 && A[n - 1][m - 2] != '-' && (A[n - 1][x] == 'K' || A[n - 1][x] == 'M' || A[n - 1][x] == 'S'))
{
    if (A[n - 1][x] == A[n - 1][m - 1])
    {
        for (i = x; i <= m - 1; i++)
        {
            A[n - 1][i] = A[n - 1][m - 1];
        }
        flag = 1;
    }
    x--;
}
```

İlk olarak dönüşüm için arada K, S, M'den biri olması gerek. Bunu while döngüsü içerisinde gösterdim. x değerini girilen m(sütün değeri)'den 3 çıkararak girilen değerden matriste 2 sol sütundan başlayarak en sol sütuna kadar kontrol ettirdim (x>=0). Flag eğer 1 olursa, sola doğru kontrol ederken kendi renginden birini bulmuş oluyor ve döngüden çıkıyor. Flag bu açıdan çok önemli. Örneğin eğer flag olmasaydı K ve K arası bütün değerler Kırmızı değerine dönüşürdü. For döngüsü ile ise aradaki değerleri aynı harfe çevirdim.

(Aynı mantık ile Sağa, Aşağıya ve Yukarıya doğru kontrol ettirdim.)

```
// sol yukariya dogru kontrol

y = n - 3; // satir
x = m - 3; // sutun
flag = 0;
while (x >= 0 && flag == 0 && y >= 0 && A[n - 2][m - 2] != '-' && (A[y][x] == 'K' || A[y][x] == 'M' || A[y][x] == 'S'))
{
    temp_x = x;
    temp_y = y;
    if (A[temp_y][temp_x] == A[n - 1][m - 1])
    {
        while (temp_y < n - 1 && temp_x < m - 1)
        {
            A[temp_y][temp_x] = A[n - 1][m - 1];
            temp_y++;
            temp_x++;
        }
        flag = 1;
    }
    y--;
    x--;
}
```

Sol yukarıya kontrol için ise aynı şekilde 2 yukarı satırdan ve 2 sol sütundan başlayarak sol yukarıya doğru satır ve sütun değerinin 0'dan büyük olana kadar kontrol ettirdim.

Ek olarak ise burada temp kullanmak zorunda kaldım çünkü örneğin x değerini ilk while döngüsünde azalttığında içerdeki while'de ise arttırdığında sonsuz döngüye giriyordu. Bu sebeple ayrı bir yere depolamak zorunda kaldım.

(Aynı şekilde sağ yukarı, sağ aşağı ve sol aşağı doğru kontrol ettirdim ve uygun harflere dönüştürdüm.)

```
    // son hali yazdir
    for (i = 0; i < boyut; i++)
    {
        for (j = 0; j < boyut; j++)
        {
            printf("%c ", A[i][j]);
        }
        printf("\n");
    }
} // end while
```

En baştaki while döngüsünün son kısmında ise tahtanın son halini yazdırdım.

```
kirmizi = 0;
mavi = 0;
sari = 0;

for (i = 0; i < boyut; i++) // tas sayilari
{
    for (j = 0; j < boyut; j++)
    {
        if (A[i][j] == 'K')
        {
            kirmizi++;
        }
        else if (A[i][j] == 'S')
        {
            sari++;
        }
        else
        {
            mavi++;
        }
    }
} // end for

    if (kirmizi > sari) // En cok bulunan tasi buldum
    {
        if (sari > mavi)
        {
            kazanan = 1;
        }
        else
        {
            if (kirmizi > mavi)
            {
                kazanan = 1;
            }
            else
            {
                kazanan = 3;
            }
        }
    }
    else
    {
        if (kirmizi > mavi)
        {
            kazanan = 2;
        }
        else
        {
            if (sari > mavi)
            {
                kazanan = 2;
            }
            else
            {
                kazanan = 3;
            }
        }
    }

    printf("OYUN BITTI.\n KIRMIZI tastan %d tane, \n SARI tastan %d tane, \n MAVI tastan %d tane bulunmaktadir. \n ", kirmizi, sari, mavi);
    printf("Kazanan %d. OYUNCU oldu! Tebrikler.", kazanan);

    return 0;
}
```

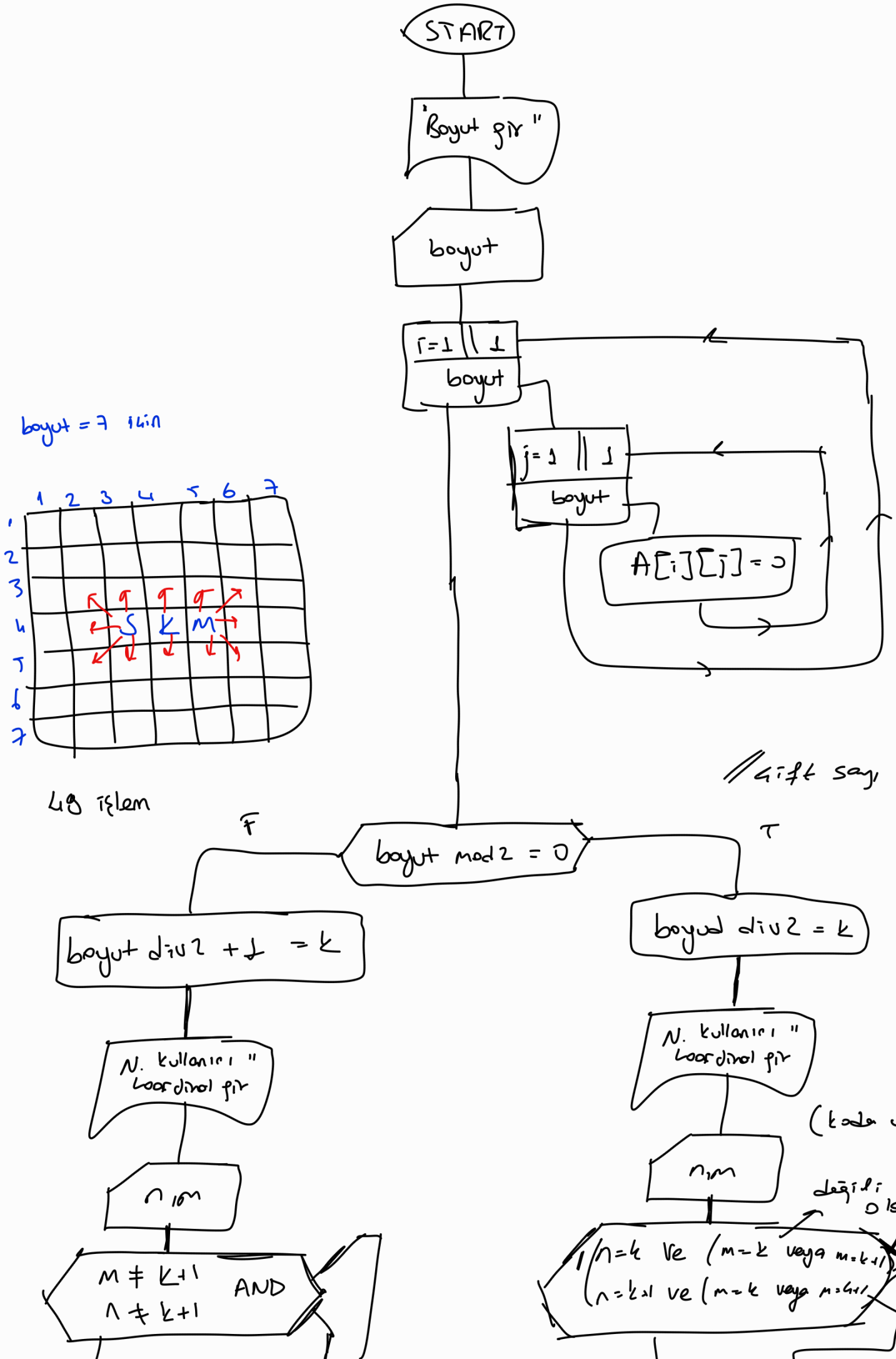
Artık kodun son kısımlarında ise tek tek kutuları kontrol ettirdim ve taşların sayısını buldum.

Son aşamada ise büyüktür küçüktür işlemleri ile en çok taşı yani kazananı buldum ve ekrana hangi taştan kaç tane olduğunu, kazananı ekrana yazdırdım.

Okuduğunuz için teşekkür ederim, Zeynel Bayhan 23011104

(Ek olarak kodu yazmaya başlamadan önce tablete yaptığım çalışmalar)

TRİVERSİ


START
"Boyut gir"
boyut
i=1 || i
boyut
j=1 || j
boyut
A[i][j] = 0
boyut = 7 için
1 2 3 4 5 6 7
1
2
3
4
5
6
7
S
K
M
Liste işlem
// Gizli sayı
F
T
boyut mod 2 = 0
boyut div 2 + 1 = k
boyut div 2 = k
N. kullanıcı " koordinat gir
N. kullanıcı " koordinat gir
(kodla yok!)
n,m
n,m
değildi
O kısın
M ≠ K+1
N ≠ K+1
AND
n=k ve (m=k veya m=k+1)
n=k+1 ve (m=k veya m=k+1)

Doğru koordinat gir.

n,m

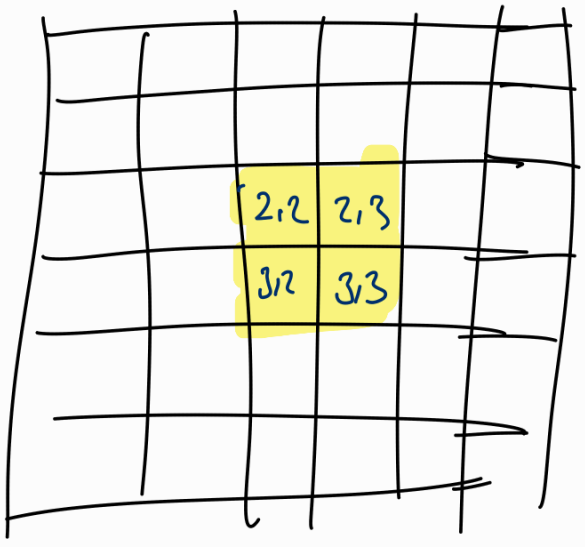
// Ortaya yerleştirdik

Doğru koordinat gir

n,m

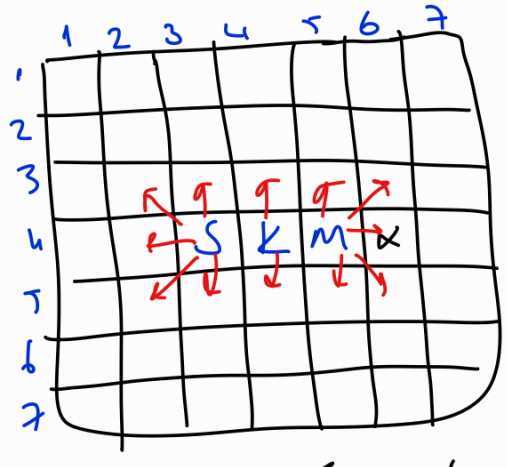
4,4

boyut = 6



4 = 3

(n,m)



6,4

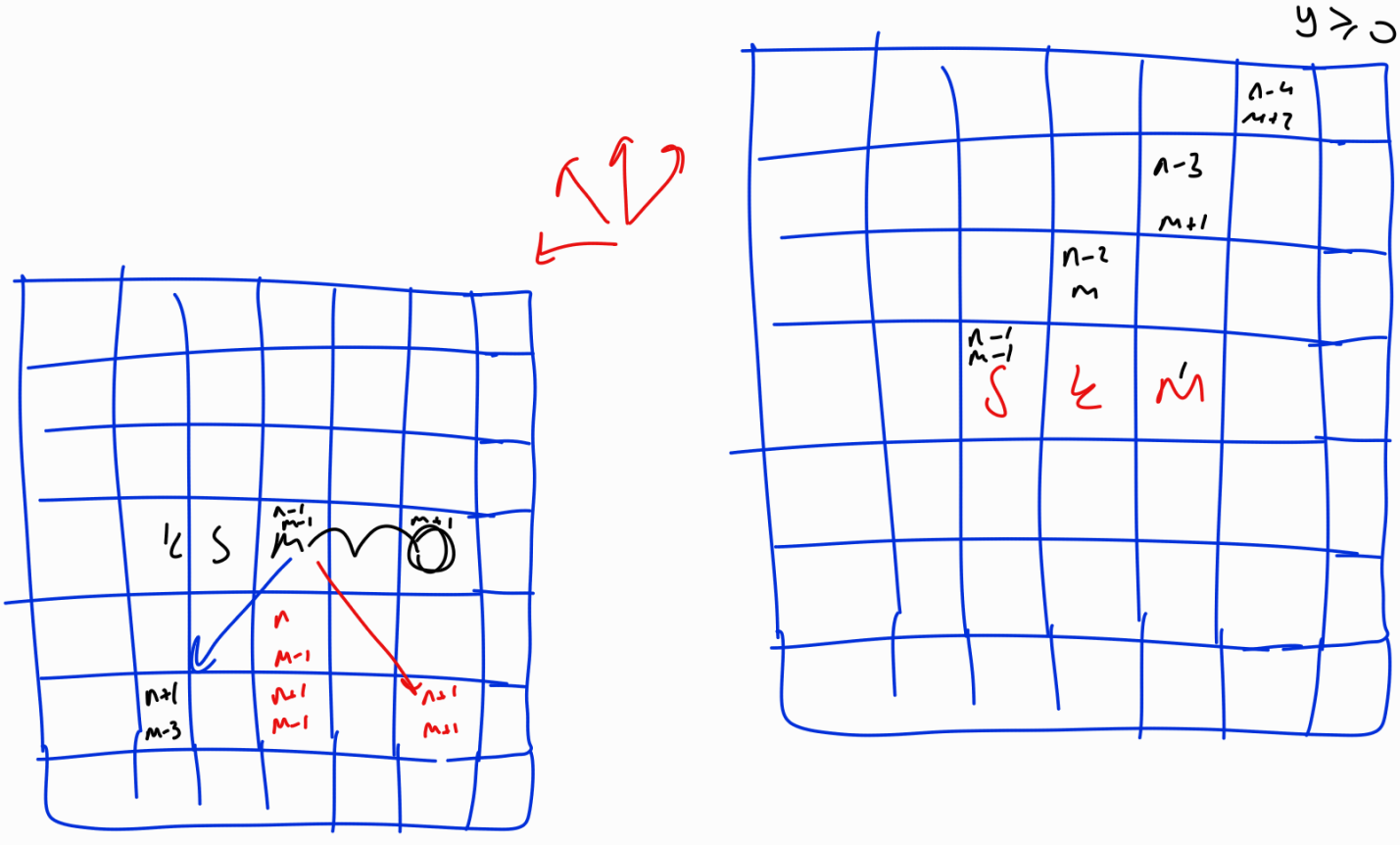
while(

ORTAYA YERLEŞTİ ✓

// Etrafına yerleştir ve kontrol et ( ).

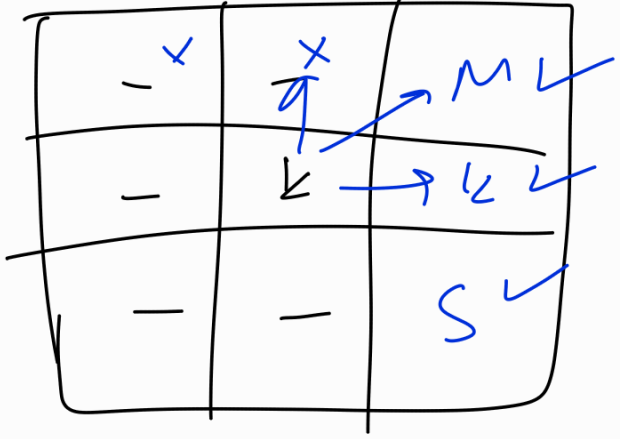
n-3=y
m+1=x

x ≤ boyut
y ≥ 0



&n &m

n-2 m-2	n-2 m-1	n-2 m
n-1 m-2	n-1 m-1	n-1 m
n m-2	n m-1	n m



Gerçeğini kontrol et!

1) Ortaya yerleştir ✓

2) Sonrakiler için etrafını kontrol et ✓

3) K M S dönüşümlerini yap ✓

4) Yazdır.