YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Bilgisayar Bilimlerine Giriş Dönem Projesi:

Triversi

Burak Başol

24011037

Öğretim Görevlisi

Dr. Öğr. Üyesi Göksel BİRİCİK

İstanbul

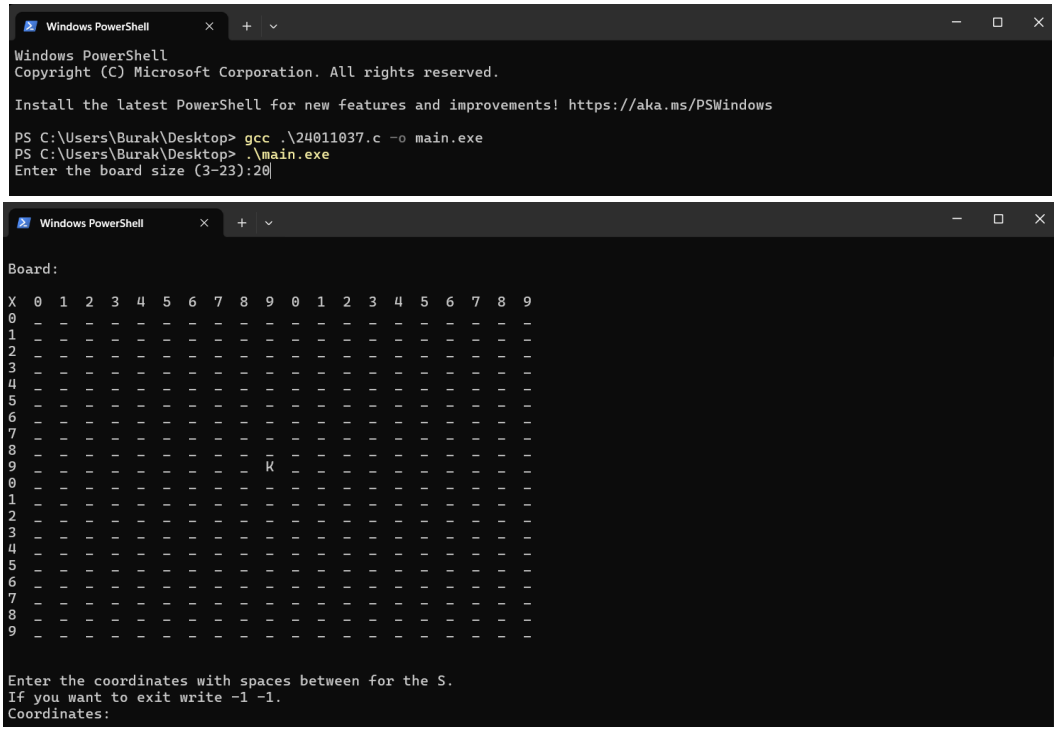
2025

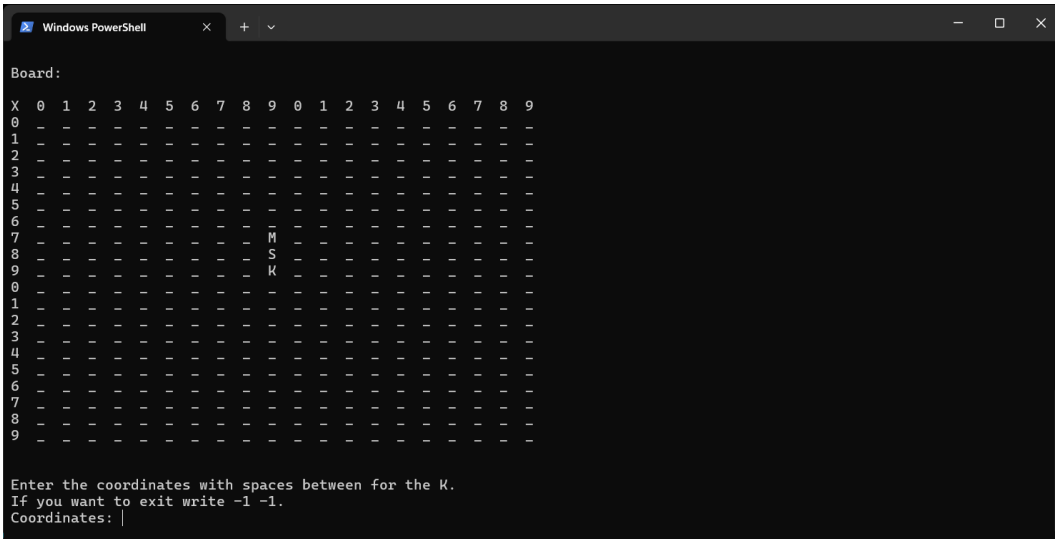
Video: https://youtu.be/Z8Kr-fgmoXk

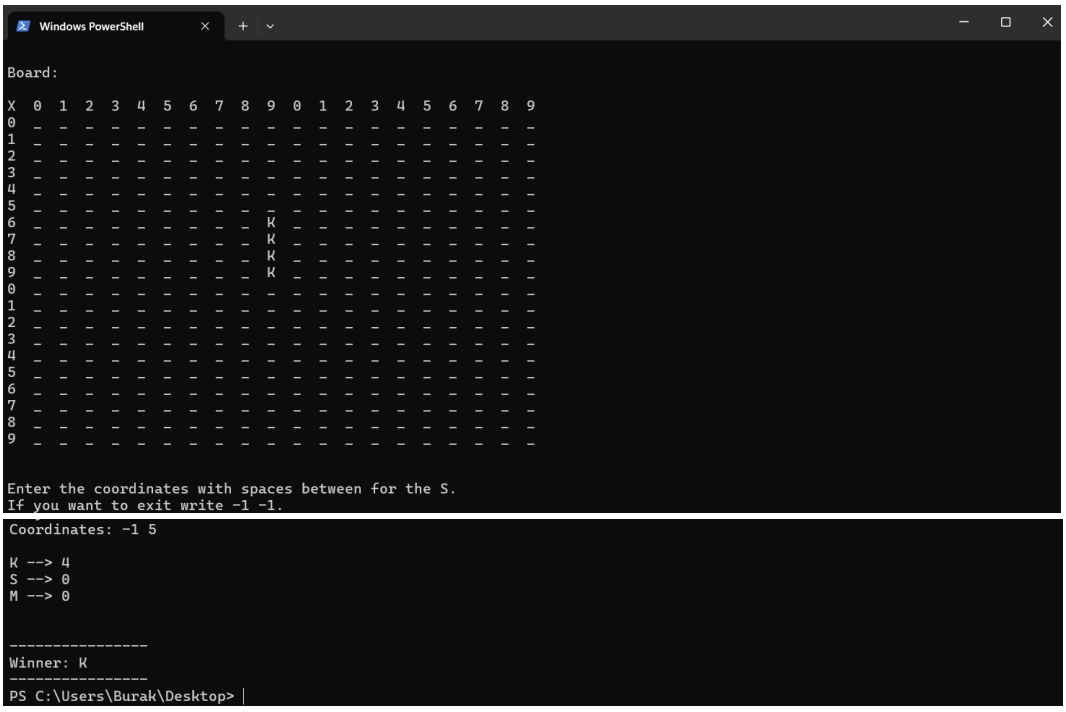
Algoritma Adımları:

1. Kullanıcıdan tahta boyutu al.
2. Sıradaki oyuncudan koordinat al.
3. Yerleştirilen taşın çevresindeki 8 bloğu kontrol et ve taş yoksa 2.adıma geri dön.
4. Yerleştirilen taşın çevresindeki 8 yöne de ilerle ve aynı renk bulana kadar devam et.
5. Aynı renkte taş bulununca aradaki taşların rengini değiştir.
6. Oyun bitince skorları ve kazananı yazdır.

Oyunun Oynanışı:




Windows PowerShell
Board:
X 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
1 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
2 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
3 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
4 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
5 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
6 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
7 _ _ _ _ _ _ _ _ _ M _ _ _ _ _ _ _ _ _
8 _ _ _ _ _ _ _ _ _ S _ _ _ _ _ _ _ _ _
9 _ _ _ _ _ _ _ _ _ K _ _ _ _ _ _ _ _ _
0 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
1 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
2 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
3 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
4 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
5 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
6 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
7 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
8 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
9 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Enter the coordinates with spaces between for the K.
If you want to exit write -1 -1.
Coordinates:


Windows PowerShell
Board:
X 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
1 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
2 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
3 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
4 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
5 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
6 _ _ _ _ _ _ _ _ _ K _ _ _ _ _ _ _ _ _
7 _ _ _ _ _ _ _ _ _ K _ _ _ _ _ _ _ _ _
8 _ _ _ _ _ _ _ _ _ K _ _ _ _ _ _ _ _ _
9 _ _ _ _ _ _ _ _ _ K _ _ _ _ _ _ _ _ _
0 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
1 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
2 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
3 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
4 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
5 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
6 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
7 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
8 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
9 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Enter the coordinates with spaces between for the S.
If you want to exit write -1 -1.
Coordinates: -1 5
K --> 4
S --> 0
M --> 0

Winner: K

PS C:\Users\Burak\Desktop>

```c
#include <stdio.h>

#define TEST

int main() {
    int i, j, k, n, x = 0, y = 0, winner = 0;
    int useless_index = 1, different_kind_of_count = 0;
    int flag = 0, _;
    int control_array[8][2] = {{-1,0},{1,0},{0,-1},{0,1},{-1,-1},{-1,1},{1,-1},{1,1}};
    int control_control_array[8] = {0};
    int useless_array[3][2] = {{'K', 0}, {'S', 0}, {'M', 0}};

    printf("Enter the board size (3-23):");

    #ifdef TEST
    n = 20;
    #else
    scanf("%d", &n);

    if ( n > 23 || n < 3) {
        printf("N value can not be bigger than 23 or smaller than 3");
        return 0;
    }
    #endif
```

- Oyunda sadece “standart input output” kütüphanesi kullanılmıştır.
- “control_array” yer değiştirme işlemleri sırasında gidilecek yönleri belirlemek için kullanılmaktadır.
- “control_control_array” oyunun orijinalinin aksine bizim oyunumuzda sadece bir satır değiştirilebilmektedir. Bu yüzden gerekli satırı bulmak isterken yapılan gereksiz işlemleri ortadan kaldırmak için bayrak görevi görmektedir.
- “useless_array” oyunculara ait renkleri ve kaç adet bulunduklarını tutmaktadır. Burada kullanılanların ve kullanılacak olanların dışında bir çok yol vardır. Ben bu şekilde daha Rahat çalışabileceğim için bu yolu tercih ettim.
- Diğer preprocess komutları üretim aşamasında kolaylık sağlaması için kullanılmışlardır ve her seferinde n değeri girilmesini engellemektedir “#define TEST” kısmı yorum satırına dönüştürülerek değer alma kısmı açılmaktadır.

```
    char board[n][n];

    for ( i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            board[i][j] = '_';
        }
    }
    board[(n / 2) - 1][(n / 2) - 1] = 'K';
```

- Bu şekilde bir uygulama yapılmasının sebebi board matrisinin boyutunun değişkene bağlı olarak tanımlanmasıdır. Sabit boyutlu bir matrise tanımlama aşamasında değer eklenebilmektedir. Bu durum bizim matrisimiz için geçerli olmadığı için 2 adet for döngüsü kullanarak sonradan değer eklenmektedir.
- Ayrıca 1.oyuncu tahtanın merkezine oynamak zorunda olduğu için otomatik olarak K harfi yerleştirilmiş olarak gelmektedir ve sıra 2.oyuncuya geçmektedir.

```
    for ( _ = n * n - 1; _ > 0 && flag == 0; _--) {

        do {
            printf("\e[1;1H\e[2J");

            printf("\nBoard:\n\nX  ");

            for ( i = 0; i < n; i++) printf("%d  ", i % 10);
            printf("\n");

            for ( i = 0; i < n; i++){
                printf("%d  ", i % 10);
                for (j = 0; j < n; j++) {
                    printf("%c  ", board[i][j]);
                }
                printf("\n");
            }
            printf("\n\nEnter the coordinates with spaces between for the %c.\n
                   If you want to exit write -1 -1.\nCoordinates: ",
                    useless_array[useless_index % 3][0]);
            scanf("%d %d", &x, &y);

            if ( x == -1 || y == -1) flag = 1;
        }
```

- For bloğu yapılabilecek hamle sayısı kadar dönmektedir. Do bloğunun içerisinde ise tahta yazdırılmakta ve oyuncudan koordinatlar istenmektedir.

```
    while (( board[x][y] != '_' ||
          ((!( x != n - 1 && y != 0)     || (board[x + 1][y - 1] == '_')) &&
           (   x == n - 1                ||  board[x + 1][y]     == '_') &&
           (!( x != n - 1 && y != n - 1) || (board[x + 1][y + 1] == '_')) &&
           (                 y == 0      || (board[x][y - 1]     == '_')) &&
           (                 y == n - 1  || (board[x][y + 1]     == '_')) &&
           (!( x != 0     && y != 0)     || (board[x - 1][y - 1] == '_')) &&
           (   x == 0                    ||  board[x - 1][y]     == '_') &&
           (!( x != 0     && y != n - 1) || (board[x - 1][y + 1] == '_')))) && flag == 0);
```

- Do bloğuna ait while döngüsünün içerisinde tahtayla ilgili kontroller yapılmaktadır. Bunlar tahtada dolu olan yere renk yerleştirilmek istenmesi ve tahtadaki taşlarla alakası olmayan bir yere blok koyulmak istenmesidir. Ayrıca flag değişkeni kontrol edilerek oyunun bitip bitmemesi gerektiği kontrol edilir.

```
        if ( flag == 0) {
            board[x][y] = useless_array[useless_index % 3][0];
            useless_index++;
        }

        for ( i = 1; i < n; i++) {
            for ( j = 0; j < 8; j++) {
                if ( ( ( (x + i * control_array[j][0]) >= 0 && (x + i * control_array[j][0]) < n) &&
                       ( (y + i * control_array[j][0]) >= 0 && (y + i * control_array[j][0]) < n)) &&
                       (control_control_array[j] == 0 &&
                        board[x + i * control_array[j][0]][y + i * control_array[j][1]] == board[x][y])) {
                    for (k = 1; k < i; k++) {
                        board[x + k * control_array[j][0]][y + k * control_array[j][1]] = board[x][y];
                    }
                    j = 8;
                    i = n;
                }
                else if ( board[x + i * control_array[j][0]][y + i * control_array[j][1]] == '_') {
                    control_control_array[j] = 1;
                }
            }
        }
        for ( i = 0; i < 8; i++) control_control_array[i] = 0;
    }
```

- Bu kısım değiştirme algoritmasına aittir. Yerleştirilen bloktan çıkarak 8 yönde aynı rengi bulmaya çalışır. Arada boşluk görürse o alanı kontrol etmeyi bırakır. Devam ettiği yerlerde ilk gelen aynı rengi alır ve arasını değiştirir. Sonrasında döngülerden çıkar.

```c
    printf("\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (board[i][j] == 'K') {
                useless_array[0][1]++;
            }
            else if (board[i][j] == 'S') {
                useless_array[1][1]++;
            }
            else if (board[i][j] == 'M') {
                useless_array[2][1]++;
            }
        }
    }
    printf("K --> %d\n", useless_array[0][1]);
    printf("S --> %d\n", useless_array[1][1]);
    printf("M --> %d\n", useless_array[2][1]);

    if ( useless_array[0][1] >= useless_array[1][1] && useless_array[0][1] >= useless_array[2][1]) {
        winner = 'K';
    }
    else if ( useless_array[1][1] >= useless_array[0][1] && useless_array[1][1] >= useless_array[2][1]) {
        winner = 'S';
    }
    else if ( useless_array[2][1] >= useless_array[1][1] && useless_array[2][1] >= useless_array[0][1]) {
        winner = 'M';
    }

    printf("\n\n----------------\nWinner: %c\n----------------", winner);
    return 0;
}
```

- Bu kısım oyunun sonunda oluşan değerlerin renklere göre hesaplanması ve kazananın belirlenmesi içindir.