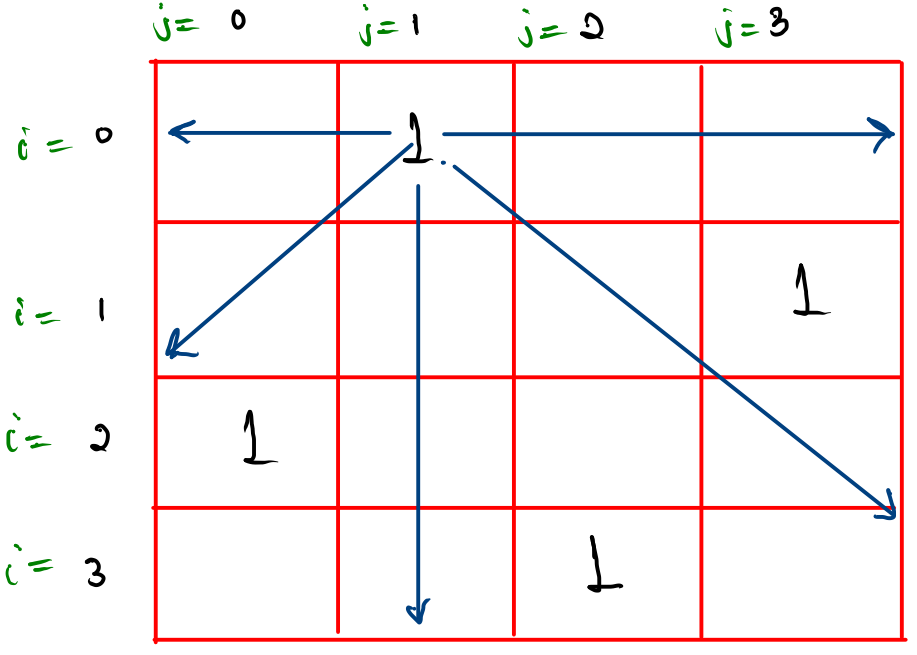
Game Theory

do what it says

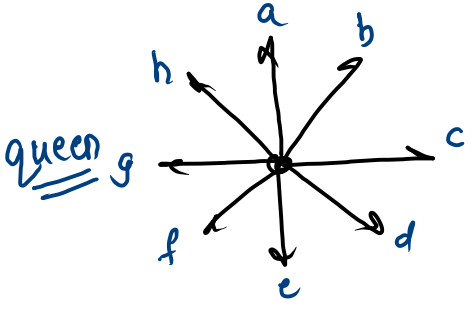
N - Queen ; long / easy

N=4



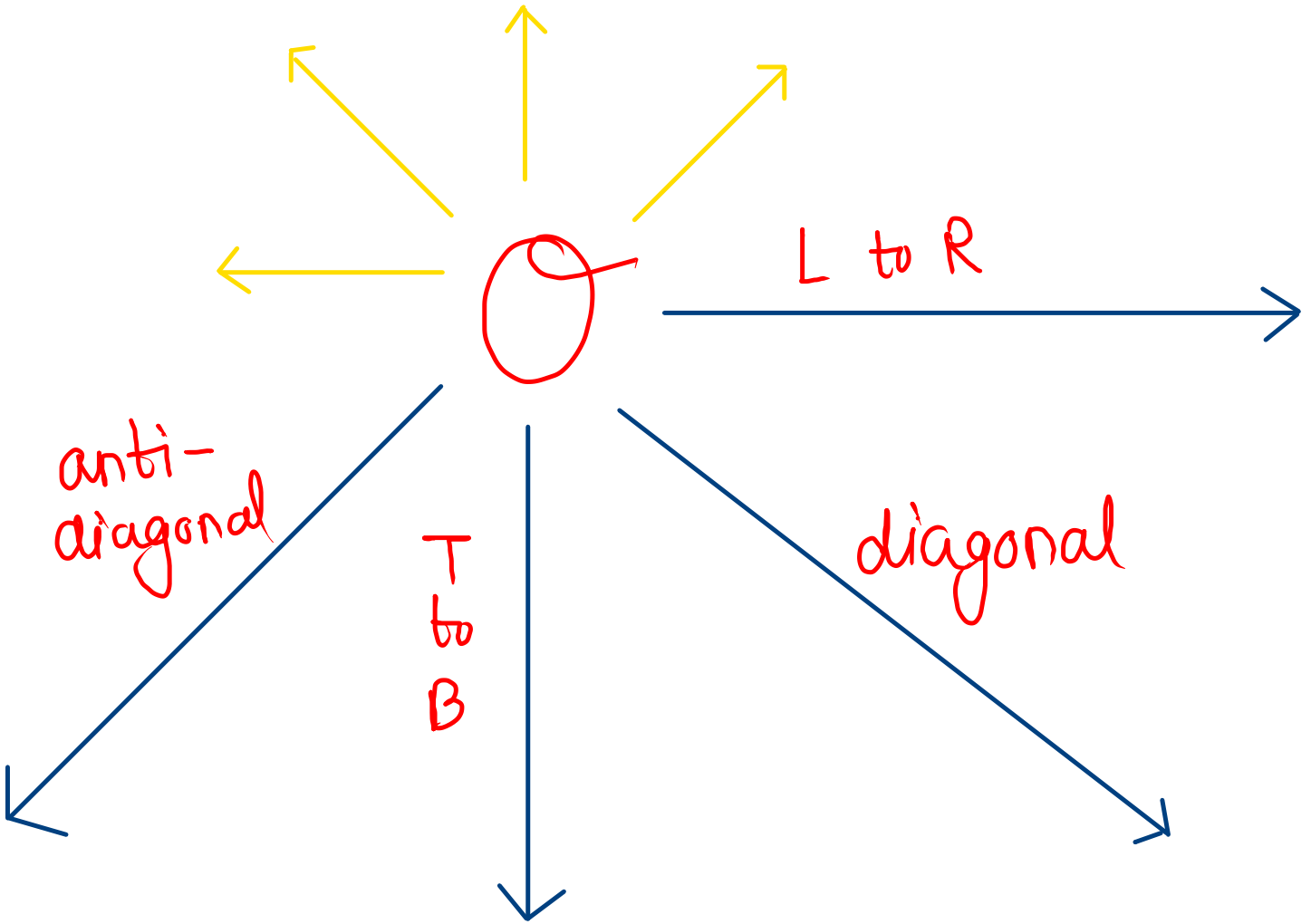
0 → nothing there

1 → queen



Queen

(i, j)



1)

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[][] arr = new int[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            arr[i][j] = scn.nextInt();
        }
    }

    nQueen(arr, n);
}
```

2)

```
public static void nQueen(int[][] arr, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if ( arr[i][j] == 1 ) {
                if ( LtoR(arr, n, i, j) == true ) {
                    System.out.println("Danger");
                    return;
                }

                if ( TtoB(arr, n, i, j) == true ) {
                    System.out.println("Danger");
                    return;
                }

                if ( diagonal(arr, n, i, j) == true ) {
                    System.out.println("Danger");
                    return;
                }

                if ( antidiagonal(arr, n, i, j) == true ) {
                    System.out.println("Danger");
                    return;
                }
            }
        }
    }
    System.out.println("N Queens");
}
```

3)

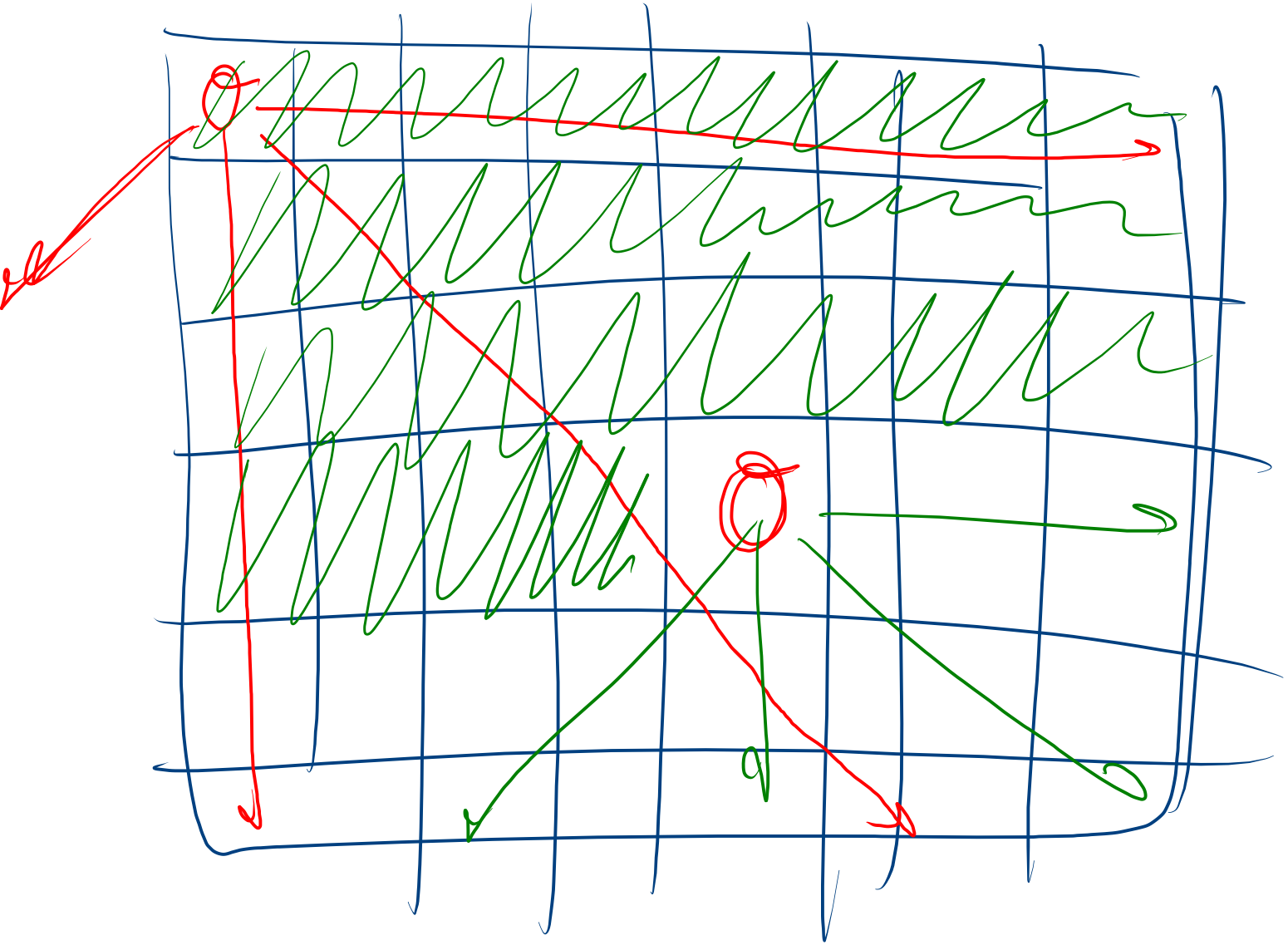
```
public static boolean LtoR(int[][] arr, int n, int i, int j) {
    j++;
    while (j < n) {
        if (arr[i][j] == 1) {
            return true;
        }
        j++;
    }
    return false;
}

public static boolean TtoB(int[][] arr, int n, int i, int j) {
    i++;
    while (i < n) {
        if (arr[i][j] == 1) {
            return true;
        }
        i++;
    }
    return false;
}
```

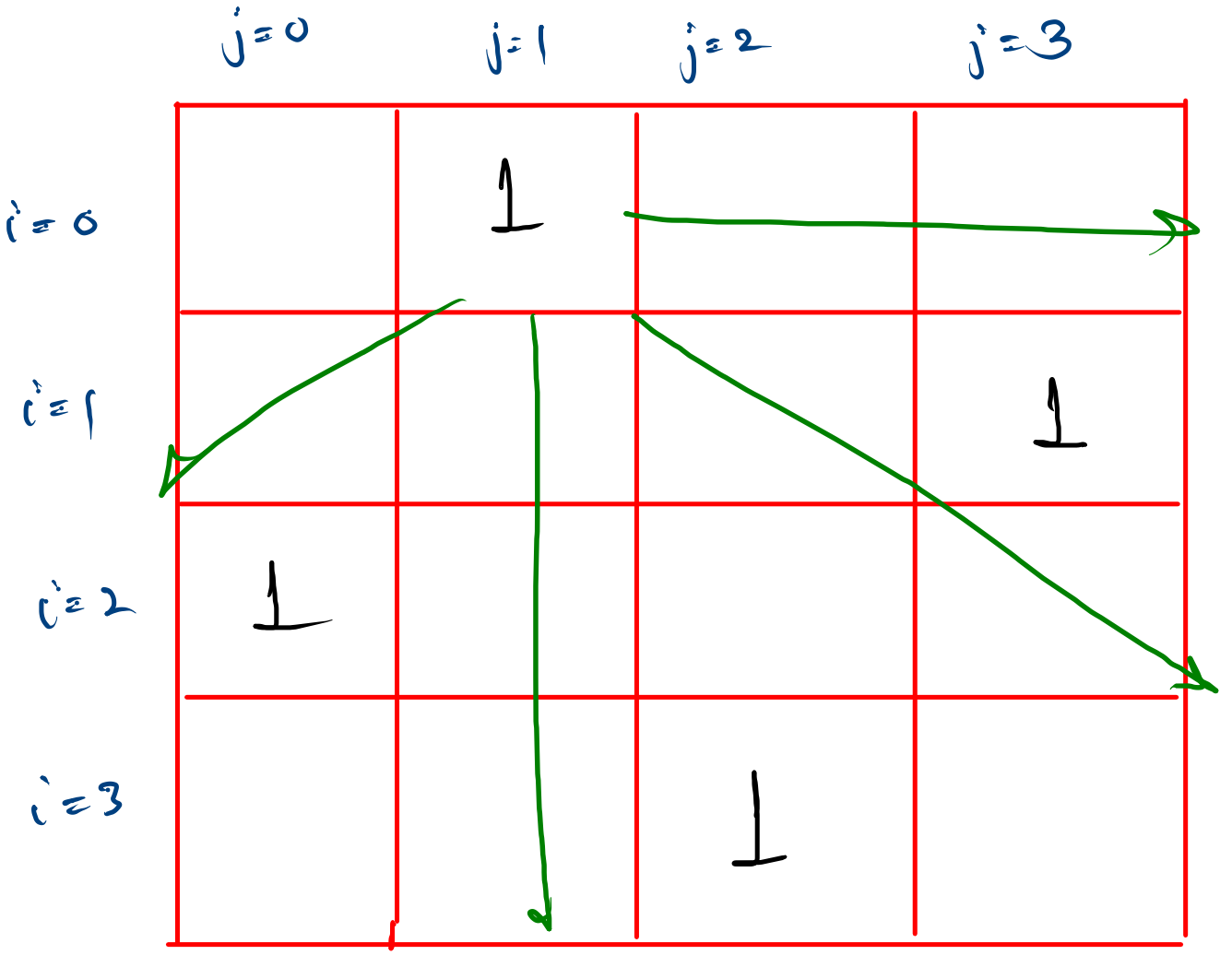
4)

```
public static boolean diagonal(int[][] arr, int n, int i, int j) {
    i++;
    j++;
    while (i < n && j < n) {
        if (arr[i][j] == 1) {
            return true;
        }
        i++;
        j++;
    }
    return false;
}

public static boolean antidiagonal(int[][] arr, int n, int i, int j) {
    i++;
    j--;
    while (i < n && j >= 0) {
        if (arr[i][j] == 1) {
            return true;
        }
        i++;
        j--;
    }
    return false;
}
```



we are already checking the previous directions so no need to check again


j=0
j=1
j=2
j=3
i=0
i=1
i=2
i=3
1
1
1
1

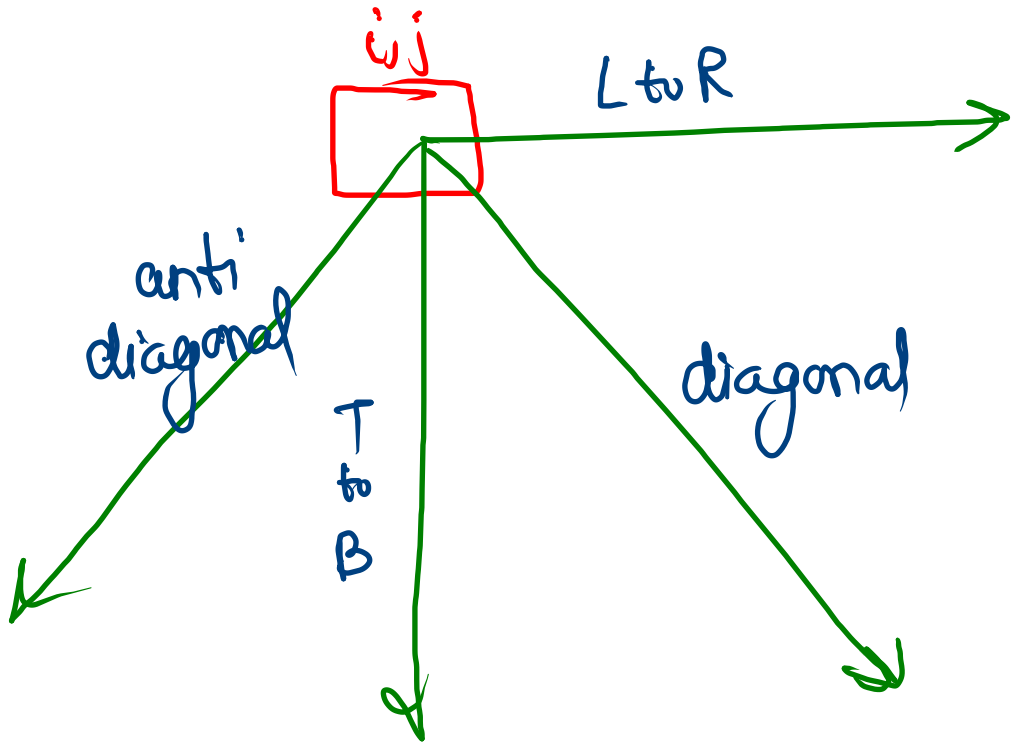
CrossWord

```
10
W C R P S R I J L J
D R T I B U B R I Y
Y E T N L P K O S T
J D S K U O O S O B
O B A F E R Y I B R
C L C B H A Q Y E O
D A W C R N L X X W
I C T E L G J N S N
Q K Q K Y E L L O W
H A O K X G R E E N
```

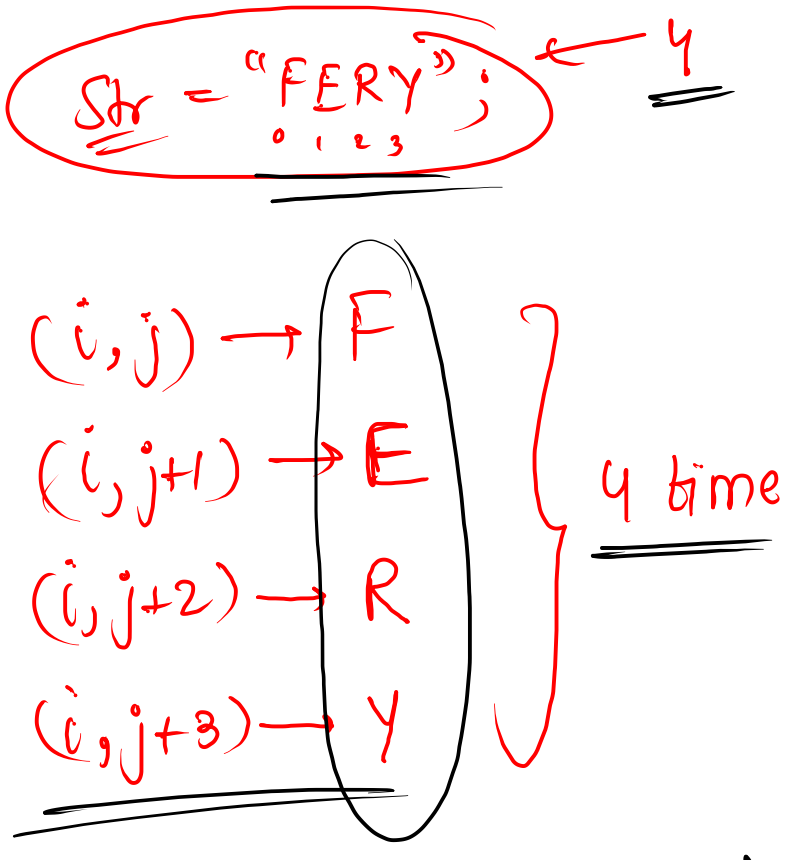
String str = "GREEN";

0 1 2 3 4

matrix



```
10
W C R P S R I J L J
D R T I B U B R I Y
Y E T N L P K O S T
J D S K U O O S O B
O B A F E R Y I B R
C L C B H A Q Y E O
D A W C R N L X X W
I C T E L G J N S N
Q K Q K Y E L L O W
H A O K X G R E E N
```



$T.C = O(N^2 * len)$

we are only moving, when words are matching

1)

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    char[][] arr = new char[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            arr[i][j] = scn.next().charAt(0);
        }
    }
    String str = scn.next();

    crossWord(arr, str, n);
}
```

2)

```java
public static void crossWord(char[][] arr, String str, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if ( arr[i][j] == str.charAt(0) ) {
                if ( LtoR(arr, n, str, i, j) == true ) {
                    System.out.println("true");
                    return;
                }

                if ( TtoB(arr, n, str, i, j) == true ) {
                    System.out.println("true");
                    return;
                }

                if ( diagonal(arr, n, str, i, j) == true ) {
                    System.out.println("true");
                    return;
                }

                if ( antidiagonal(arr, n, str, i, j) == true ) {
                    System.out.println("true");
                    return;
                }
            }
        }
    }
    System.out.println("false");
}
```

3)

```java
public static boolean LtoR(char[][] arr, int n, String str, int i, int j) {
    int count = 0, ans = 0;
    while (j < n && i < n && count < str.length()) {
        if ( arr[i][j] == str.charAt(count) ) {
            ans++;
        }
        count++;
        j++;
    }

    if (ans == str.length()) {
        return true;
    }
    return false;
}

public static boolean TtoB(char[][] arr, int n, String str, int i, int j) {
    int count = 0, ans = 0;
    while (j < n && i < n && count < str.length()) {
        if ( arr[i][j] == str.charAt(count) ) {
            ans++;
        }
        count++;
        i++;
    }

    if (ans == str.length()) {
        return true;
    }
    return false;
}
```

4)

```java
public static boolean diagonal(char[][] arr, int n, String str, int i, int j) {
    int count = 0, ans = 0;
    while (j < n && i < n && count < str.length()) {
        if ( arr[i][j] == str.charAt(count) ) {
            ans++;
        }
        count++;
        i++;
        j++;
    }

    if (ans == str.length()) {
        return true;
    }
    return false;
}

public static boolean antidiagonal(char[][] arr, int n, String str, int i, int j) {
    int count = 0, ans = 0;
    while (j >= 0 && i < n && count < str.length()) {
        if ( arr[i][j] == str.charAt(count) ) {
            ans++;
        }
        count++;
        i++;
        j--;
    }

    if (ans == str.length()) {
        return true;
    }
    return false;
}
```

```java
public static boolean LtoR(char[][] arr, int n, String str, int i, int j) {
    int count = 0, ans = 0;
    while (j < n && i < n && count < str.length()) {
        if ( arr[i][j] == str.charAt(count) ) {
            ans++;
        }
        count++;
        j++;
    }

    if (ans == str.length()) {
        return true;
    }
    return false;
}
```

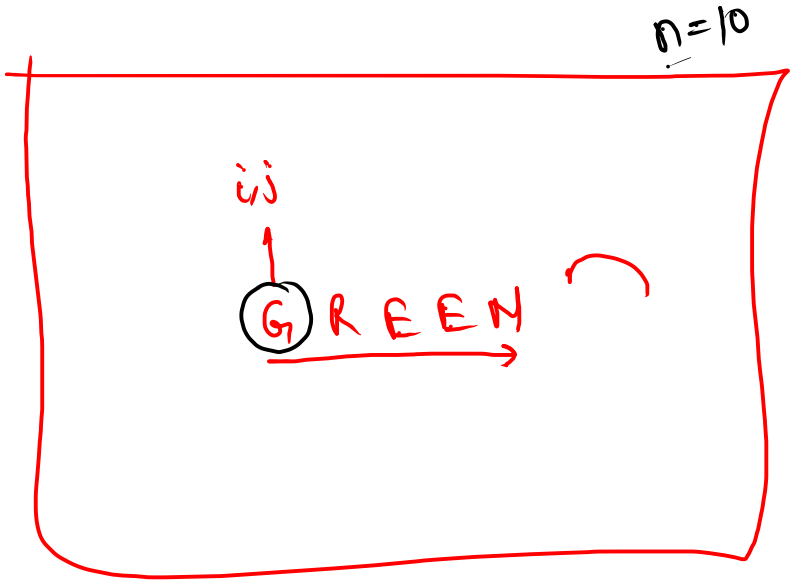
str = "GREEN" len = 5

0 1 2 3 4

count = 0 1 2 3 4 5

ans = 0 1 2 3 4 5

i = 1, j = 1



(i=1, j=1)

(1,1) → G

(1,2) → R

(1,3) → E

(1,4) → E

(1,5) → N

(1,6)

(G == G) T

(R == R) T

(E == E) T

(E == E) T

(N == N) T