44. Wildcard Matching

Given an input string (`s`) and a pattern (`p`), implement wildcard pattern matching with support for `'?'` and `'*'`.

```
'?' Matches any single character.
'*' Matches any sequence of characters (including the empty sequence).
```

The matching should cover the **entire** input string (not partial).

Note:

- `s` could be empty and contains only lowercase letters `a-z`.
- `p` could be empty and contains only lowercase letters `a-z`, and characters like `?` or `*`.

Example 1:

```
Input:
s = "aa"
p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input:
s = "aa"
p = "*"
Output: true
Explanation: '*' matches any sequence.
```

Example 3:

```
Input:
s = "cb"
p = "?a"
Output: false
Explanation: '?' matches 'c', but the second letter is 'a', which does not
match 'b'.
```

Example 4:

```
Input:
s = "adceb"
p = "*a*b"
Output: true
Explanation: The first '*' matches the empty sequence, while the second
'*' matches the substring "dce".
```

Example 5:

```
Input:
s = "acdcb"
p = "a*c?b"
Output: false
```

Intution:
we will create a 2d dp matrix of s.length() * p.length()
we will solve this question by dynamic programming
dp[i][j] represents wether substring[0...i-1] of p mathces with substring [0...j-1] of s

look at code you will understand clearly,

CODE:

```
bool isMatch(string s, string p) {
        int m = p.length();
        int n = s.length();
     bool dp[m+1][n+1];
        dp[0][0] = true;
        for(int i=1;i<=n;i++)
        {
           dp[0][i] = false;
        }
        for(int i=1;i<=m;i++){
           if(p[i-1] == '*')
              dp[i][0] = dp[i-1][0];
           else
              dp[i][0] = false;
        }
        for(int i=1;i<=m;i++)
           for(int j=1;j<=n;j++){
              if(p[i-1] == '*')
                 dp[i][j] = dp[i-1][j] || dp[i][j-1] || dp[i-1][j-1];
              else if(p[i-1] == '?' || p[i-1] == s[j-1])
                 dp[i][j] = dp[i-1][j-1];
              else
                 dp[i][j] = false;
           }
        return dp[m][n];
    }
```