10. Regular Expression Matching

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

```
'.' Matches any single character.
'*' Matches zero or more of the preceding element.
```

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 = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it
becomes "aa".
```

Example 3:

```
Input:
s = "ab"
p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".
```

Example 4:

```
Input:
s = "aab"
p = "c*a*b"
Output: true
Explanation: c can be repeated 0 times, a can be repeated 1 time. Therefore, it matches "aab".
```

Example 5:

```
Input:
s = "mississippi"
p = "mis*is*p*."
Output: false
```

INTUTION:

we will create a dp[][] array

where dp[i][j] denotes whether the string s[0...i] matches with p[0...j]

let m = s.length()

n = p.length()

create dp[m+1][n+1] = {false};

set dp[0][0] = true   // '' matches with ''

fill 0th row   // dp[0][i] = dp[0][i-2] if(s[i-1] == '*') else dp[0][i] = false

meaning 'a*','a*b*' => all evaluate to true

also set dp[i][0] = false

now let's fill the matrix

```
dp[i][j] = {dp[i-1][j-1]    ; s[i-1] = p[j-1] OR p[j-1] = '.'
           {if(p[i-1] == '*') => dp[i][j-2]    ; if(s[i-1] = p[j-2] || p[j-2] = '.')   => dp[i][j]
||  dp[i-1][j]
            {false
```

for doubts watch : https://www.youtube.com/watch?v=l3hda49XcDE

CODE:

```
bool isMatch(string s, string p) {
        int m = s.length();
        int n = p.length();
     bool dp[m+1][n+1] = {false};
        dp[0][0] = true;
        for(int i=1;i<=n;i++){
            if(p[i-1] == '*')
                dp[0][i] = dp[0][i-2];
            else
                dp[0][i] = false;
        }
        for(int i=1;i<=m;i++)
            dp[i][0] = false;
        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                if(s[i-1] == p[j-1] || p[j-1] == '.')
                    dp[i][j] = dp[i-1][j-1];
                else if(p[j-1] == '*'){
                    dp[i][j] = dp[i][j-2];
                    if(s[i-1] == p[j-2] || p[j-2] == '.')
                        dp[i][j] = dp[i][j] || dp[i-1][j];
                }
                else
                    dp[i][j] = false;
            }
        }
        return dp[m][n];
    }
```