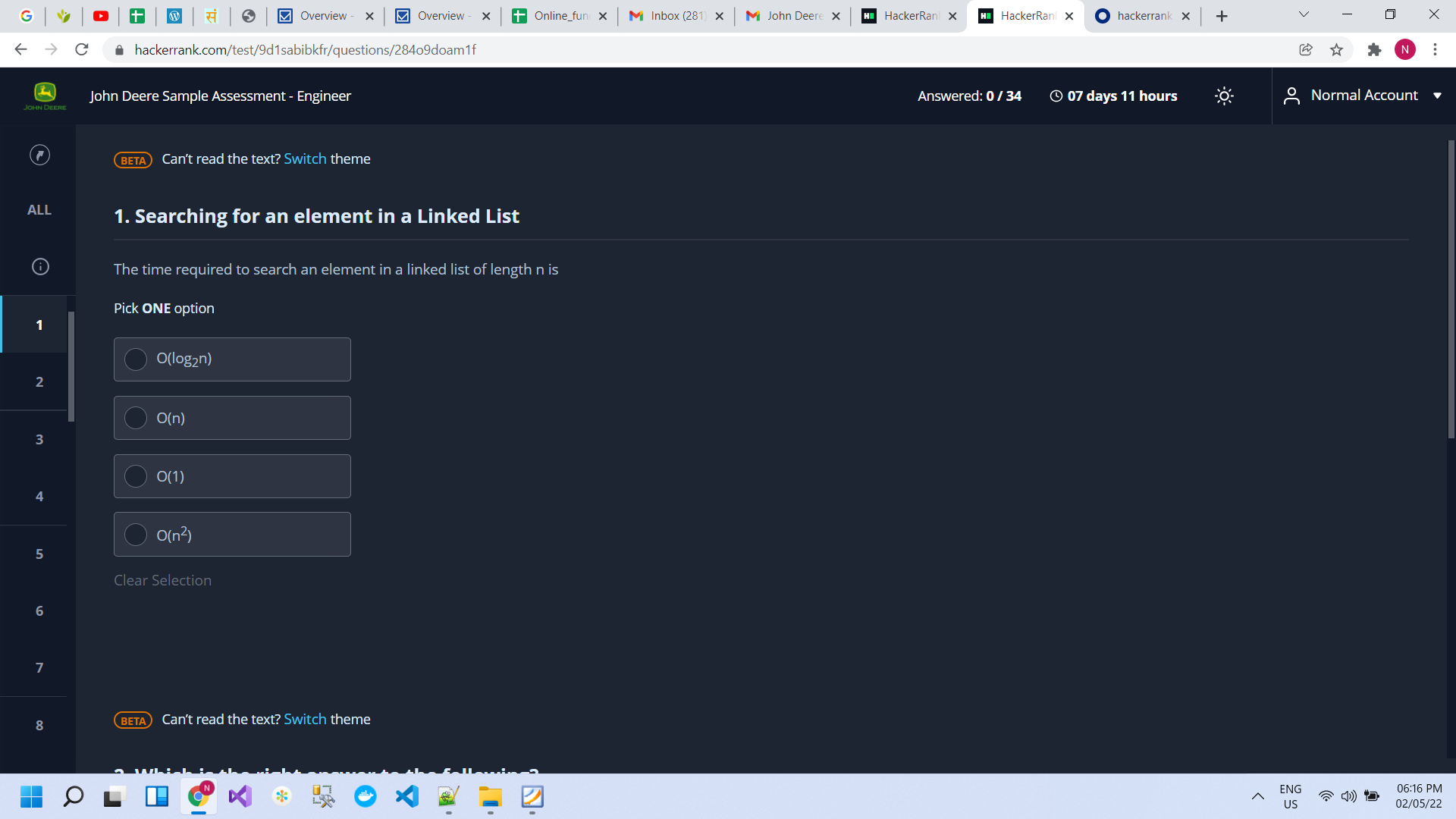
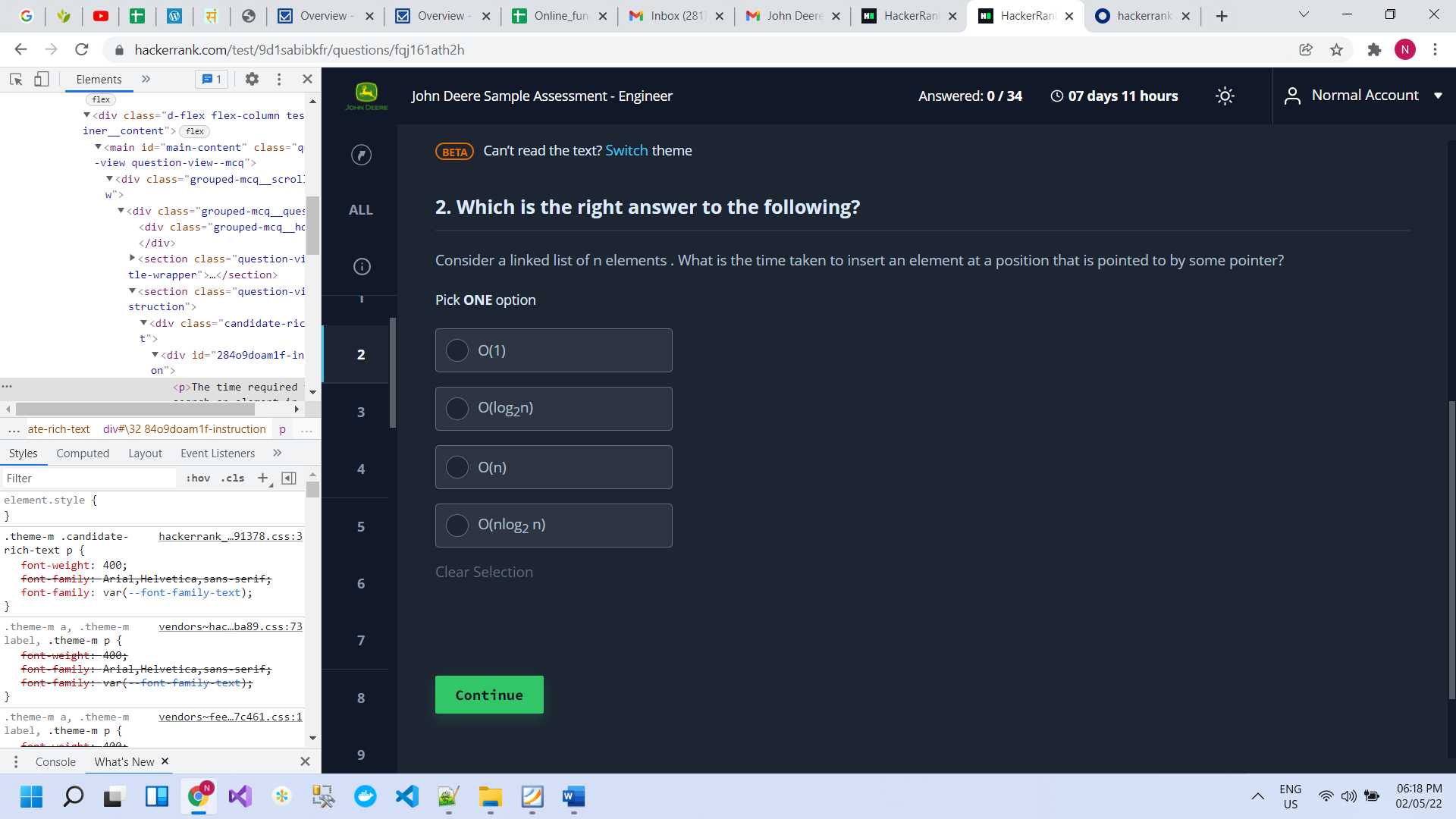
# S1

The time required to search an element in a linked list of length n is

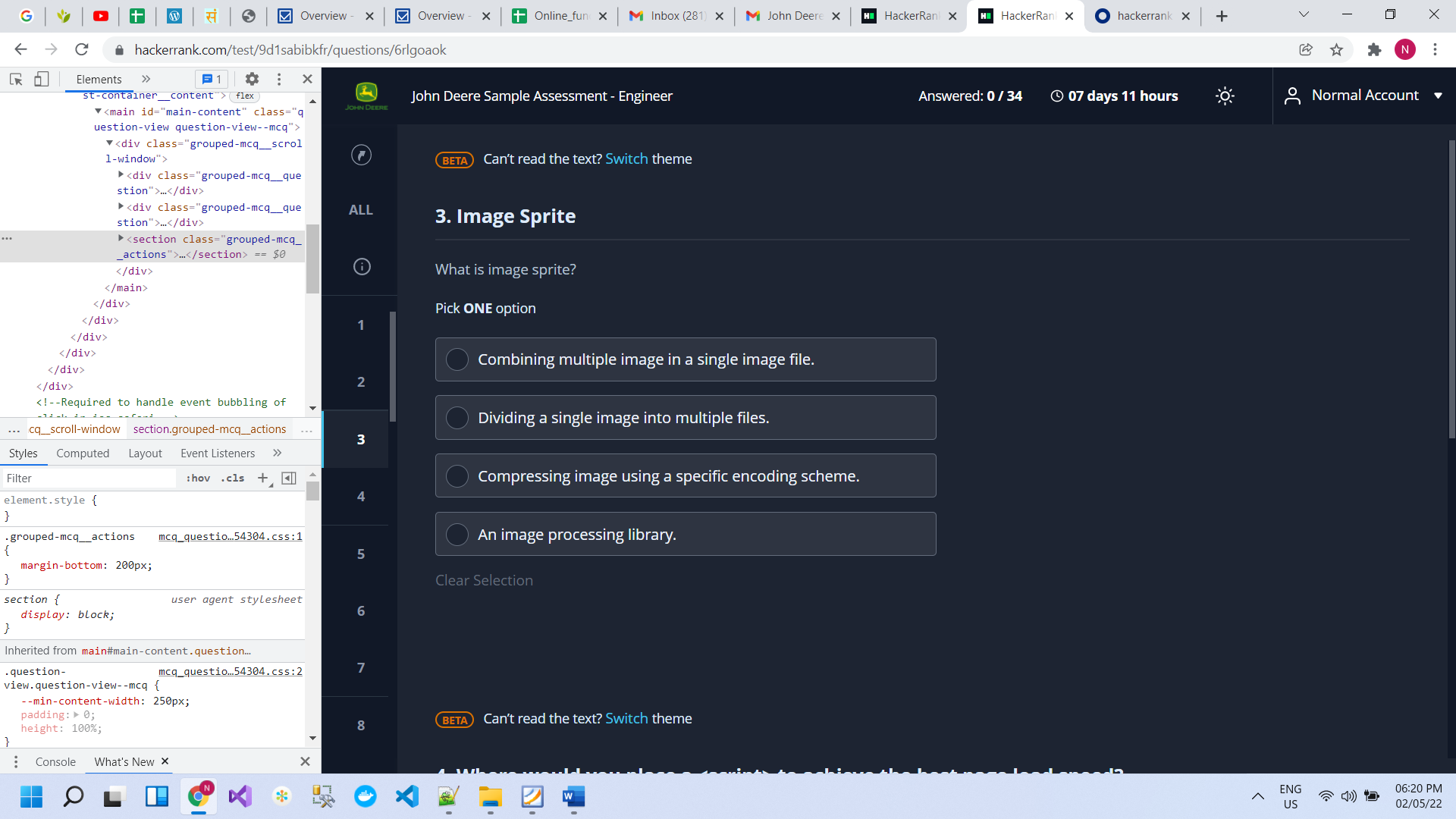


Consider a linked list of n elements . What is the time taken to insert an element at a position that is pointed to by some pointer?

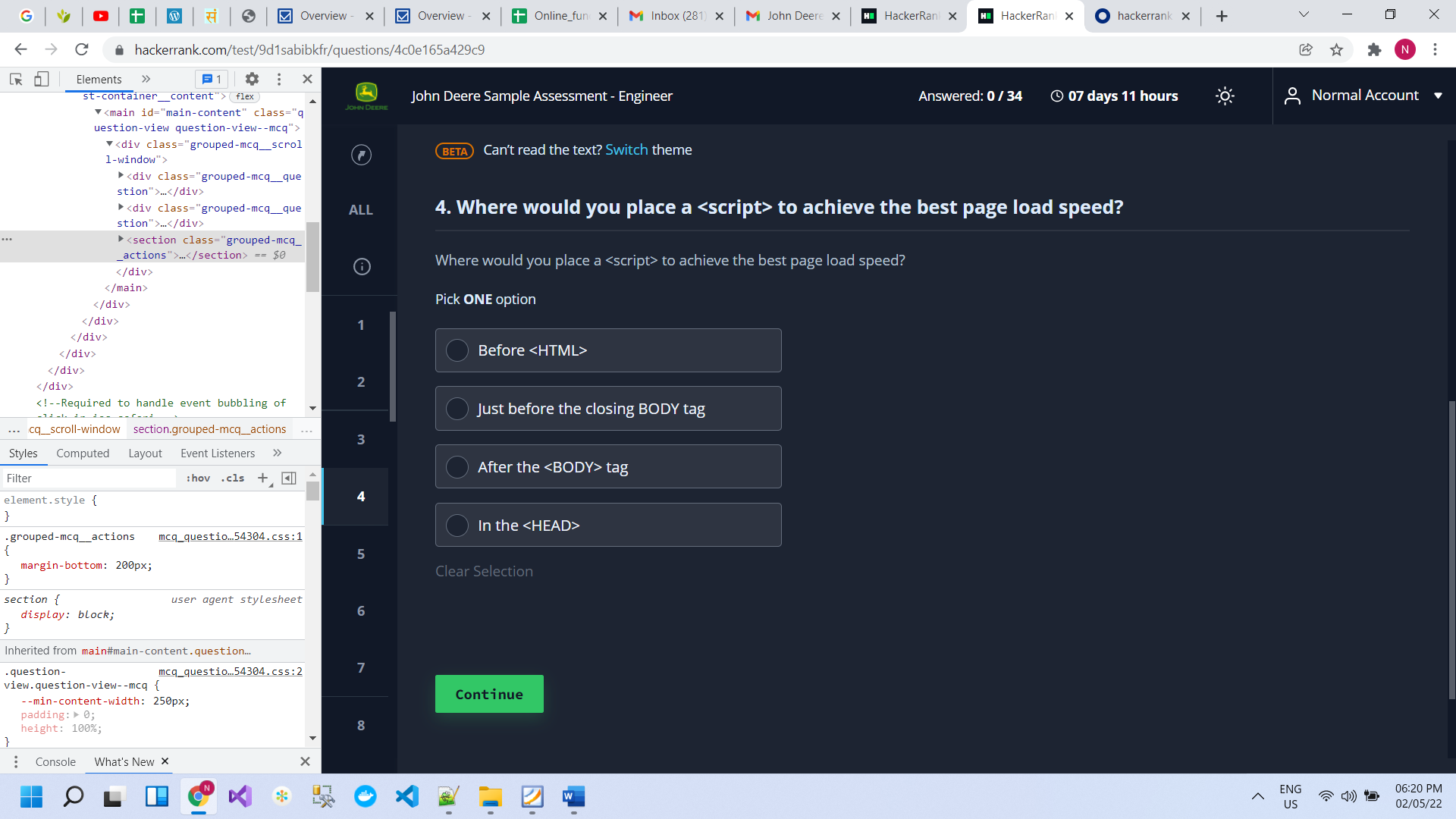


# S2

# S3



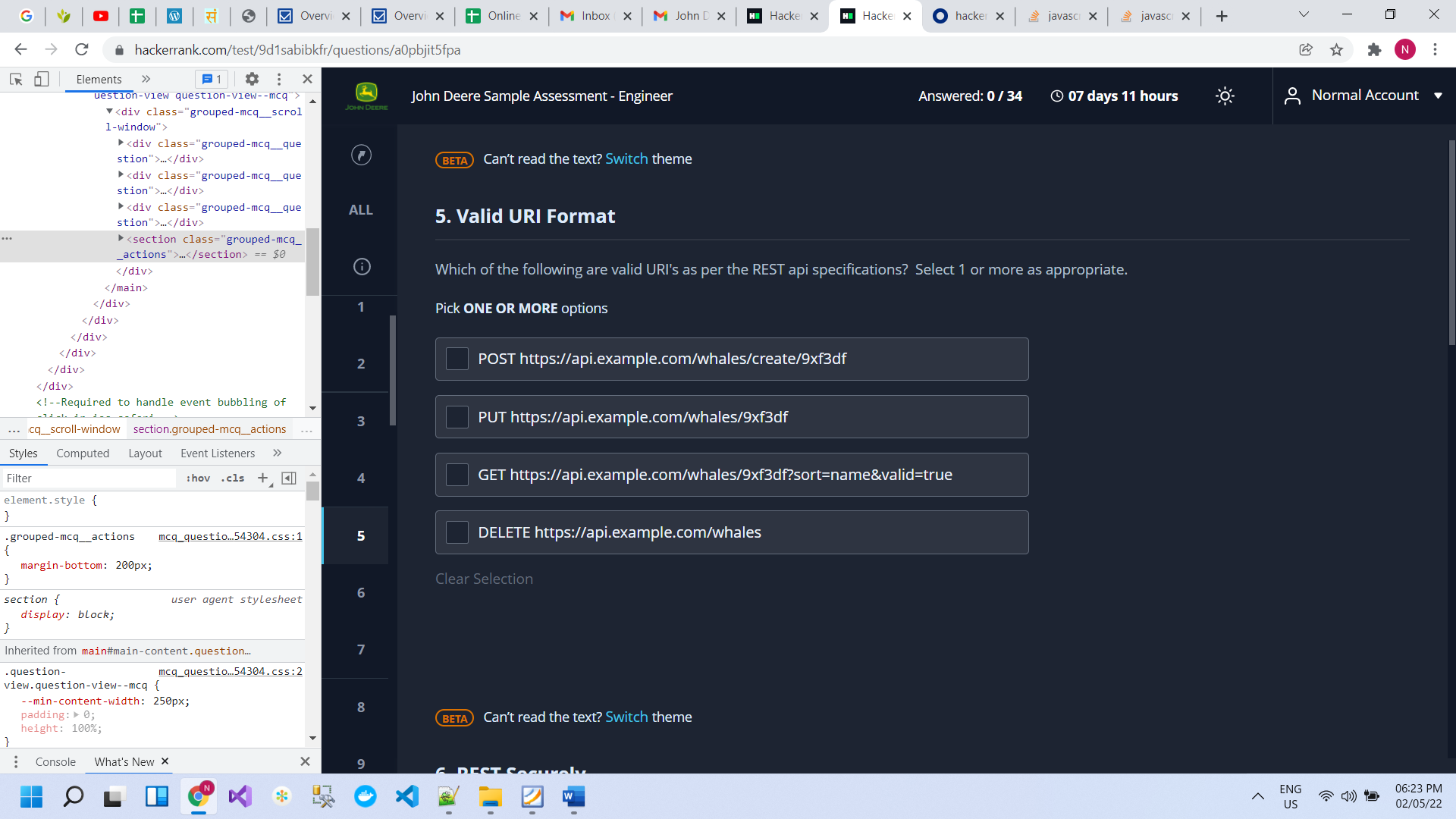
Where would you place a <script> to achieve the best page load speed?



# S4

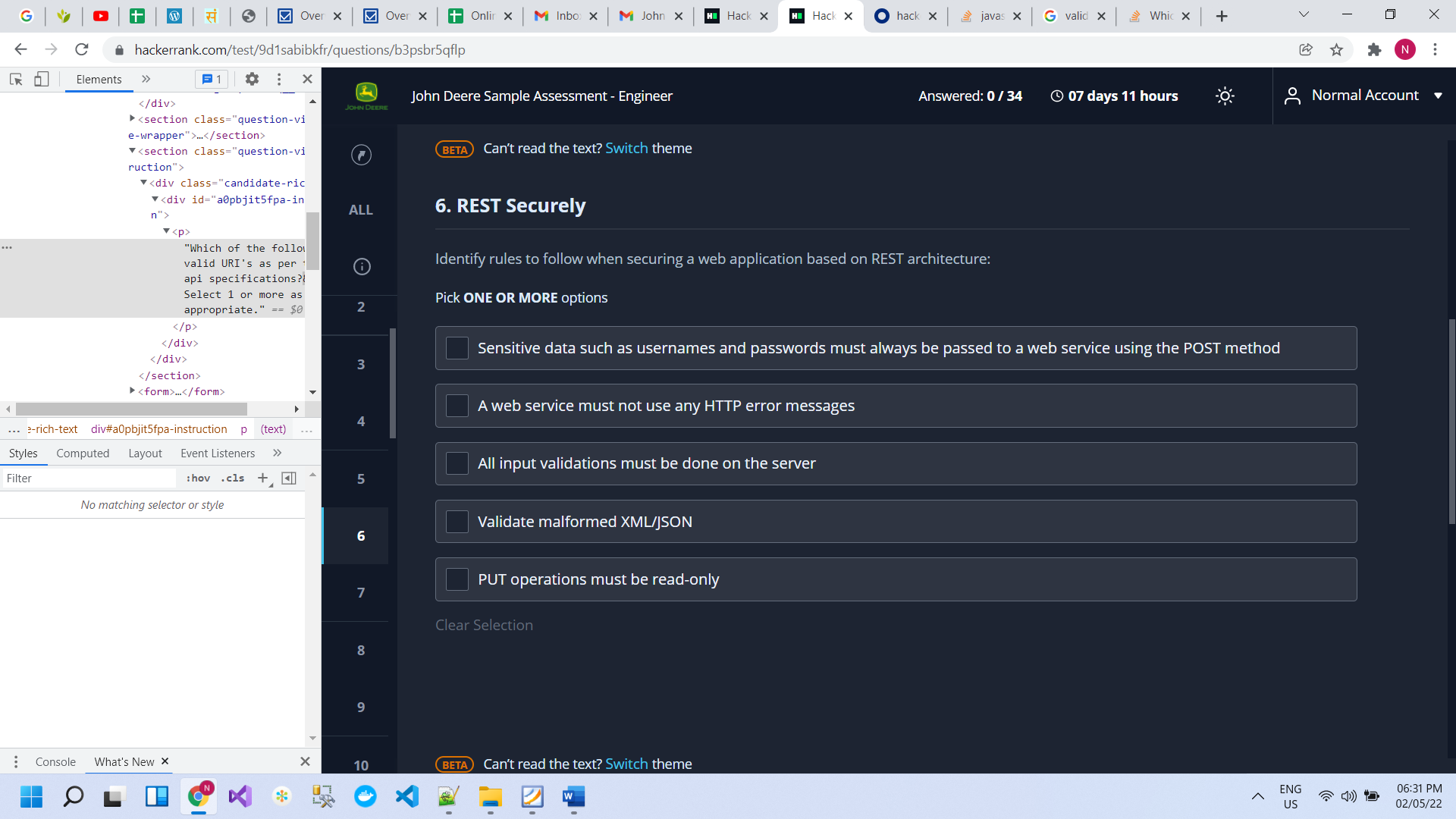
# S5

Which of the following are valid URI's as per the REST api specifications?  Select 1 or more as appropriate.

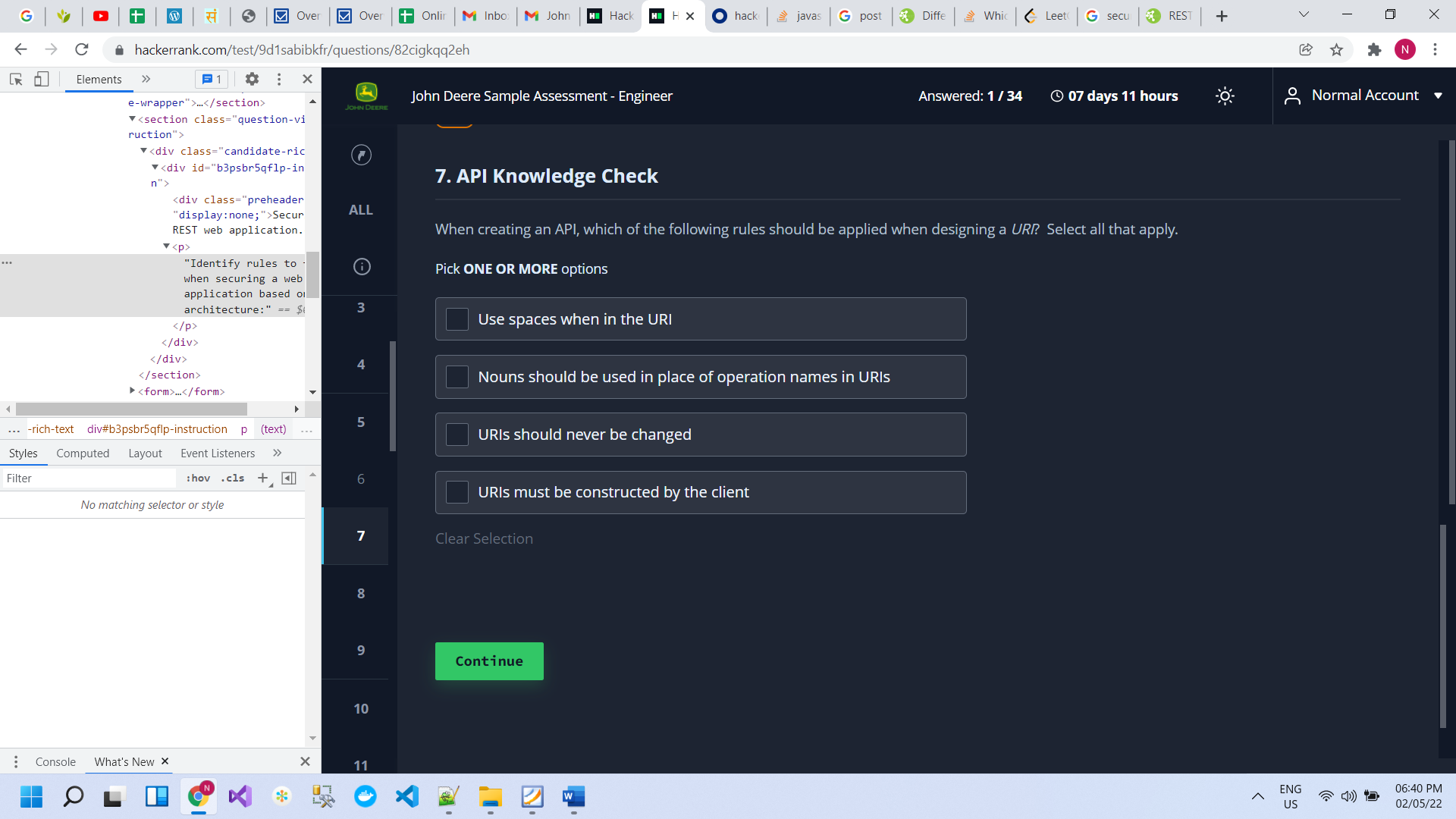


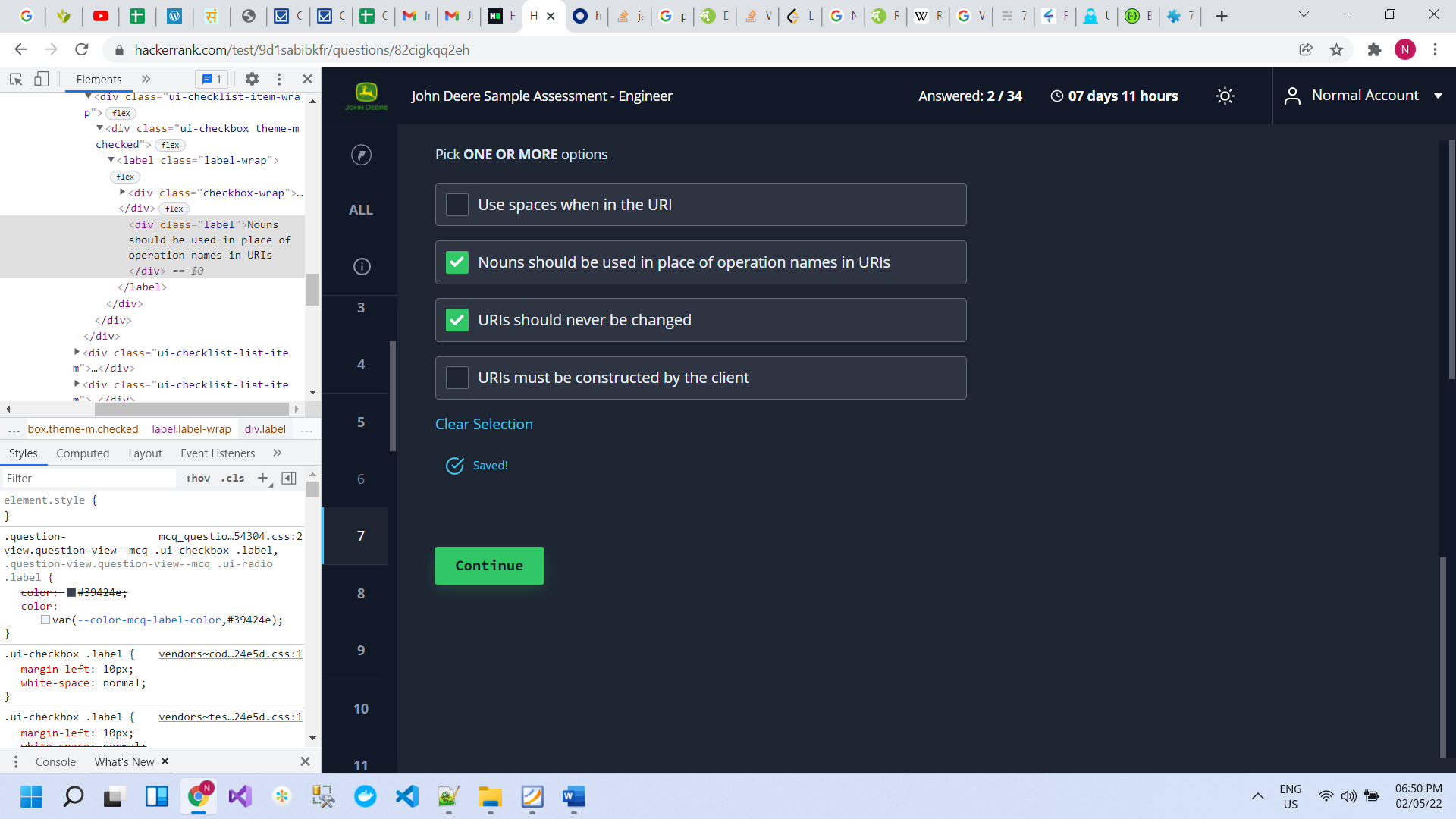
# S6

Identify rules to follow when securing a web application based on REST architecture:



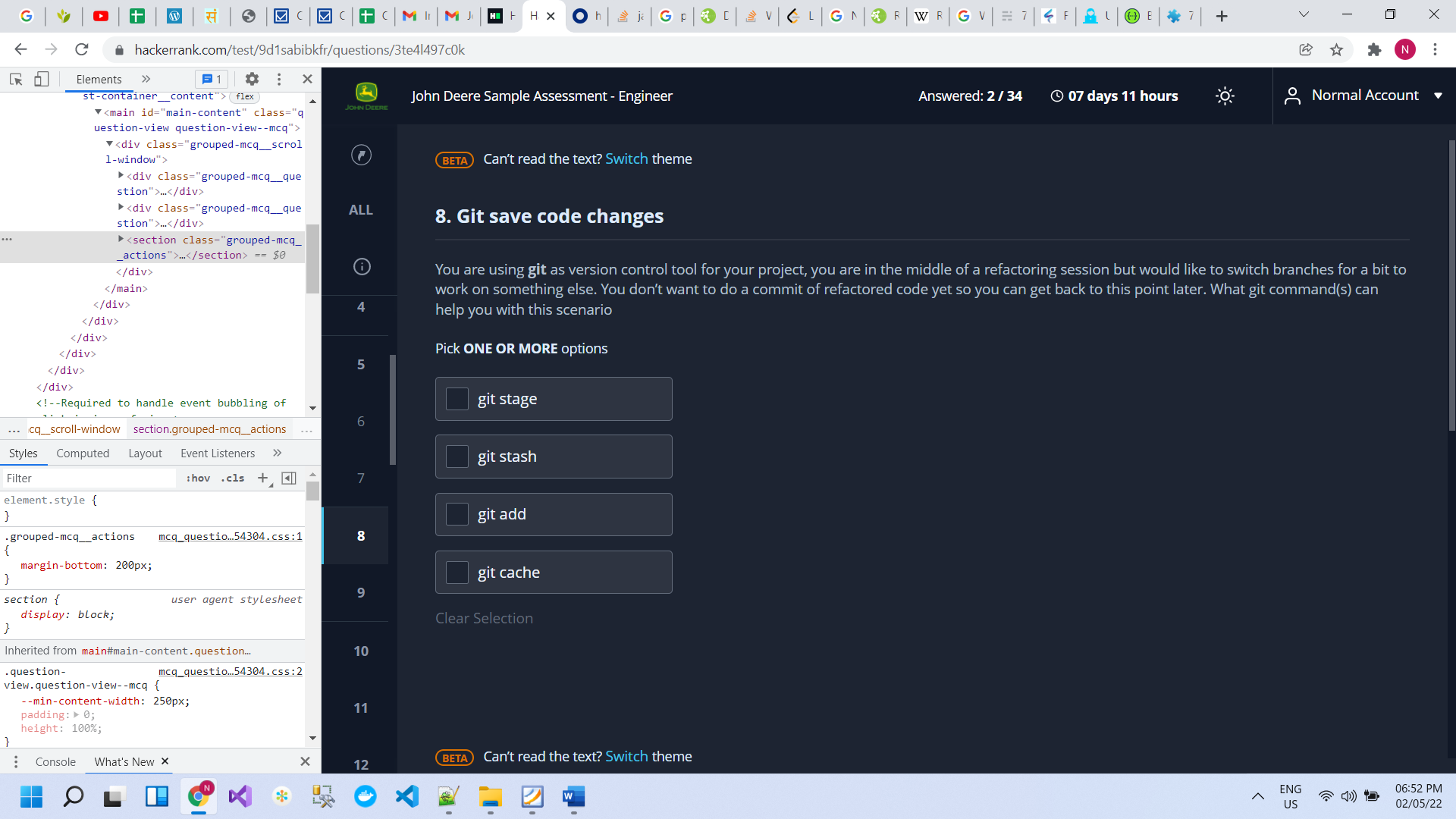
When creating an API, which of the following rules should be applied when designing a URI



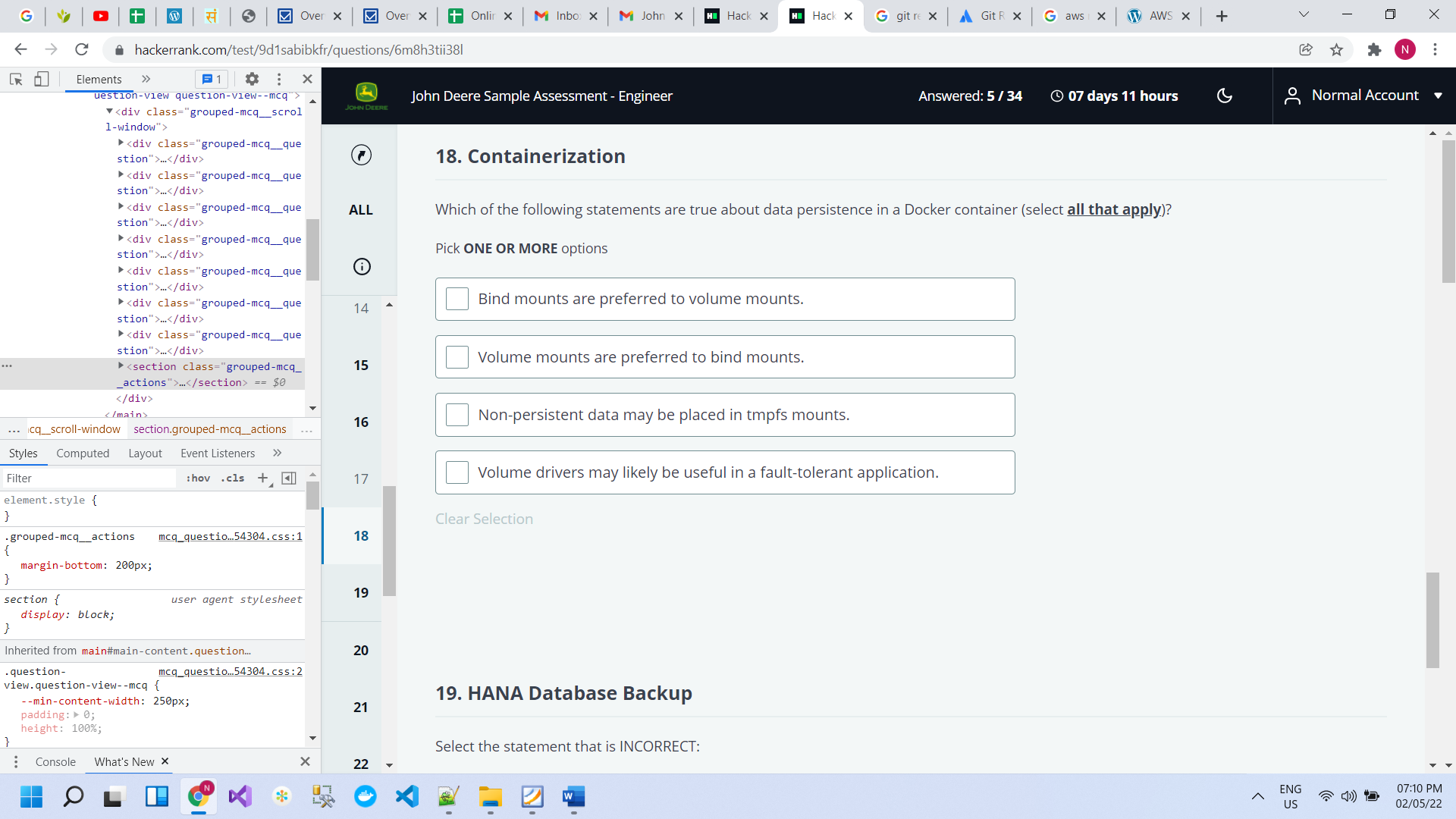


# S7

Please select the git command that will undo the last commit but will leave/retain the changes made in the last commit



# S8



# S9

# S10

# S11

# S12

# S13

# S14

# SQL

1. <p>Write a query to find Student 213's test score

select max(score) as t from student

where score not in (

select top 212 score  from student

order by score desc );

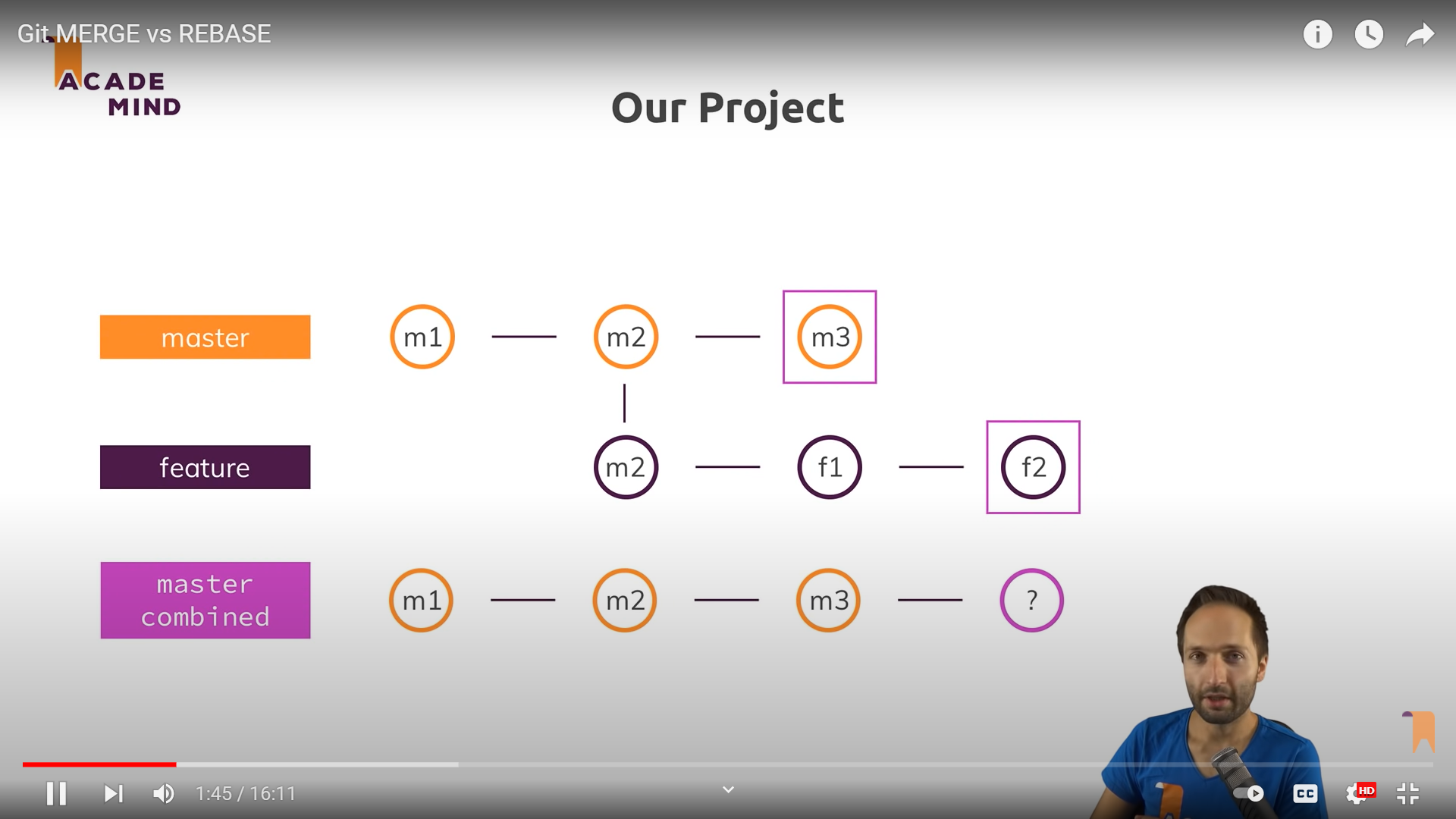
1. how to find duplicate rows in sql server

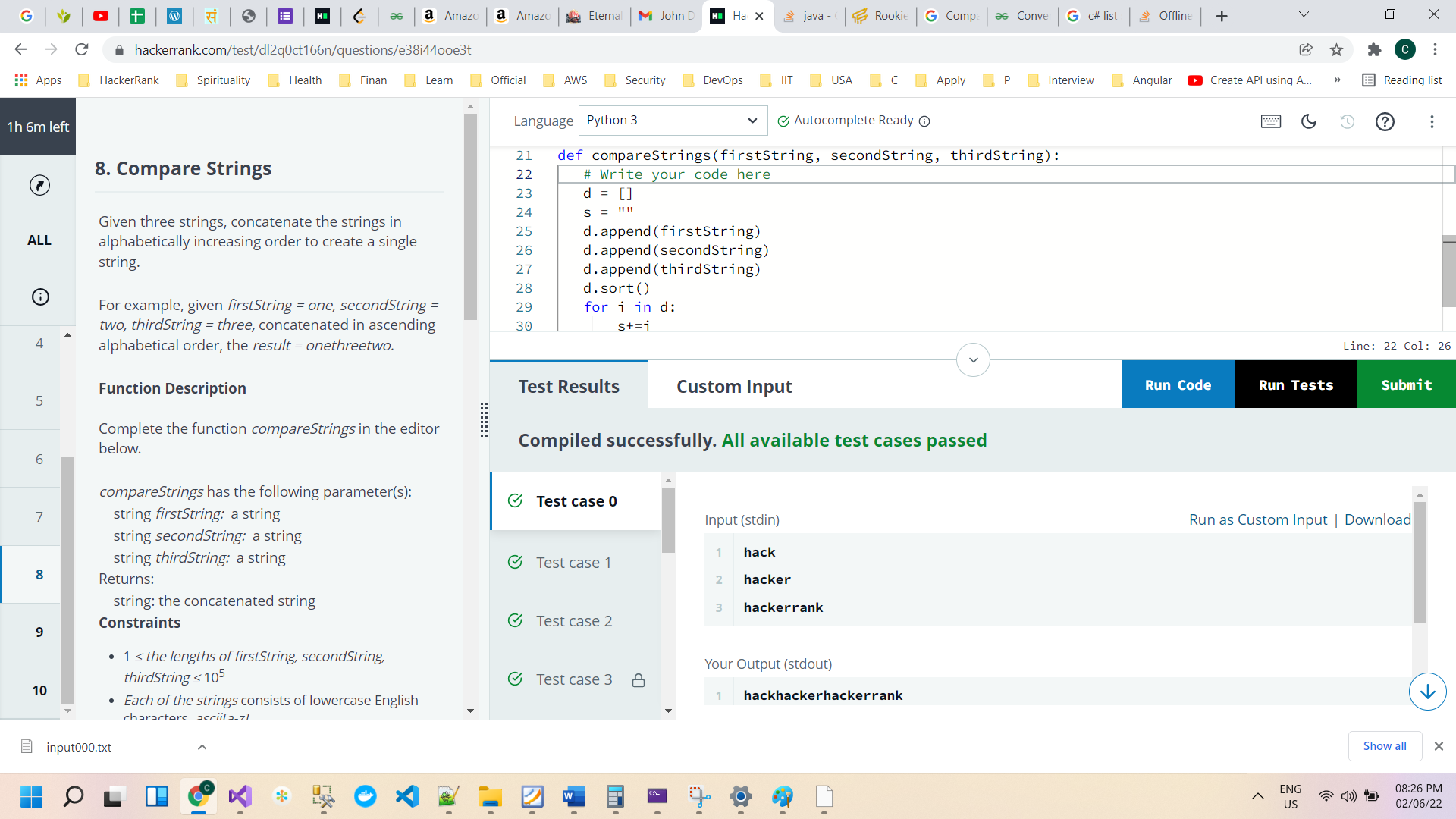
SELECT a, b, COUNT(\*) occurrences

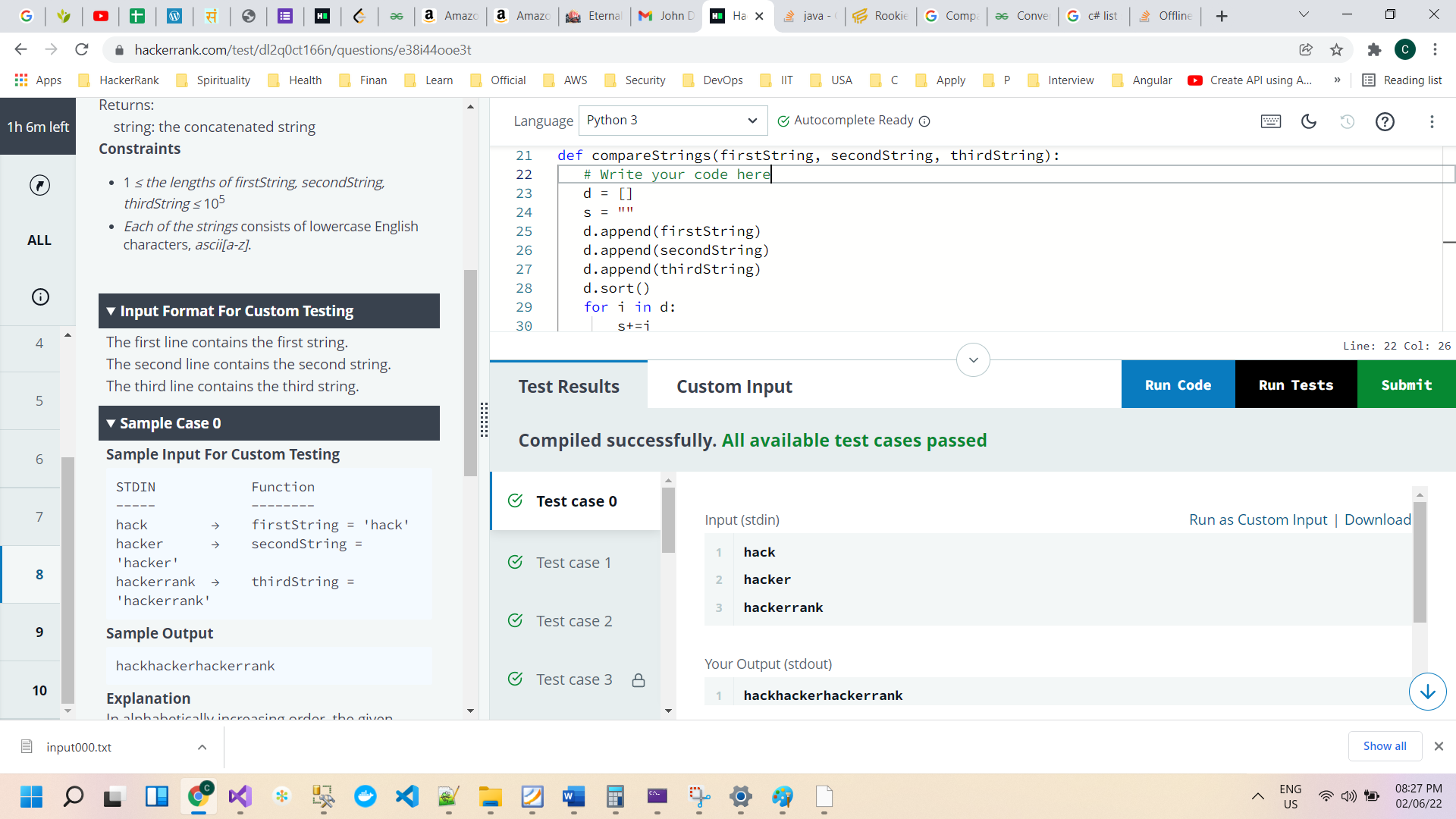
FROM t1

GROUP BY a, b

HAVING COUNT(\*) > 1;







d = []

s = ""

d.append(firstString)

d.append(secondString)

d.append(thirdString)

d.sort()

for i in d:

s+=i

return s

C#

public static string compareStrings(string firstString, string secondString, string thirdString)

    {

            List<string> lstVal = new List<string>();

            var finalResult = "";

            lstVal.Add(firstString);

            lstVal.Add(secondString);

            lstVal.Add(thirdString);

            lstVal.Sort();

            foreach (var i in lstVal)

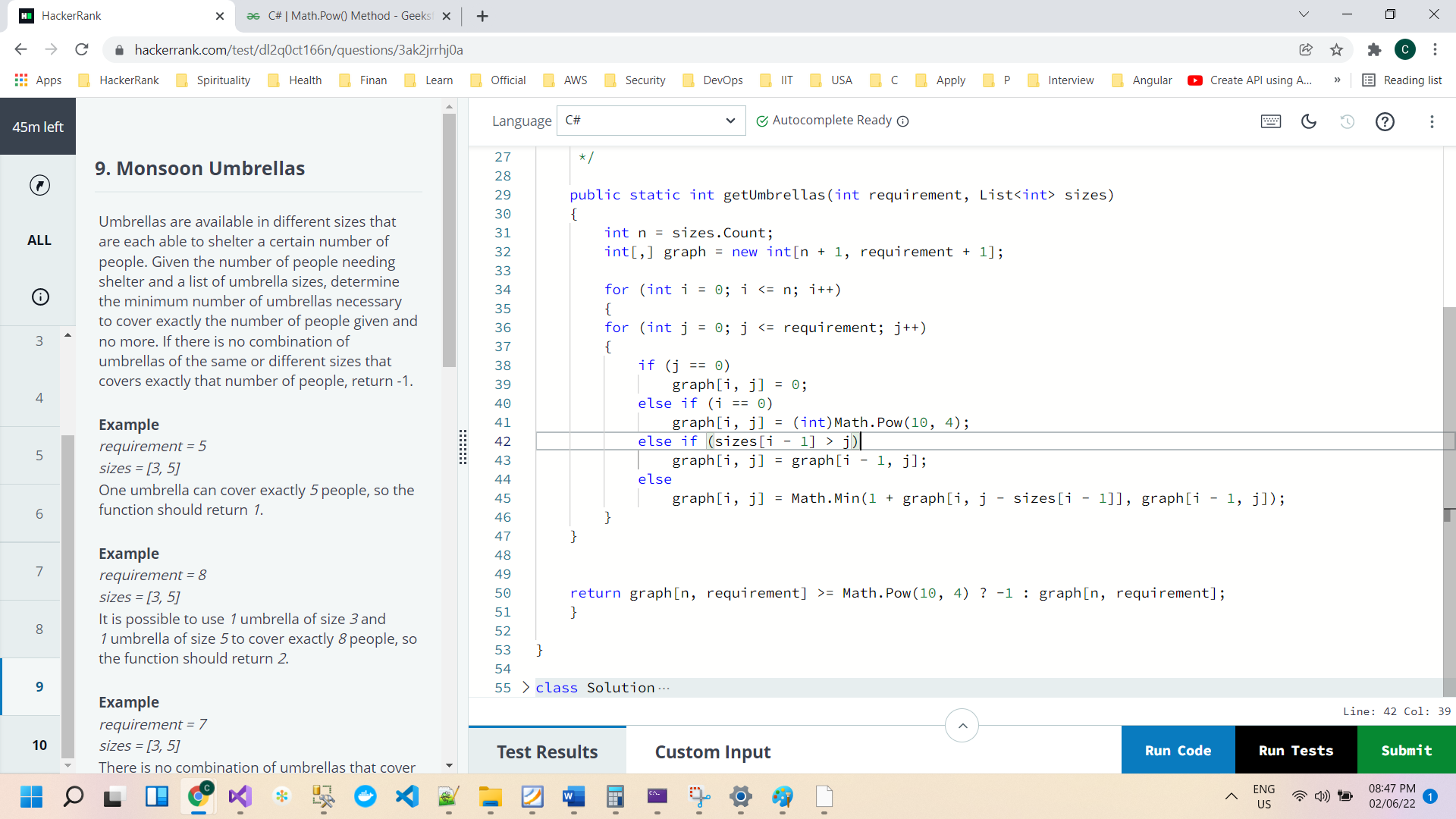
            {

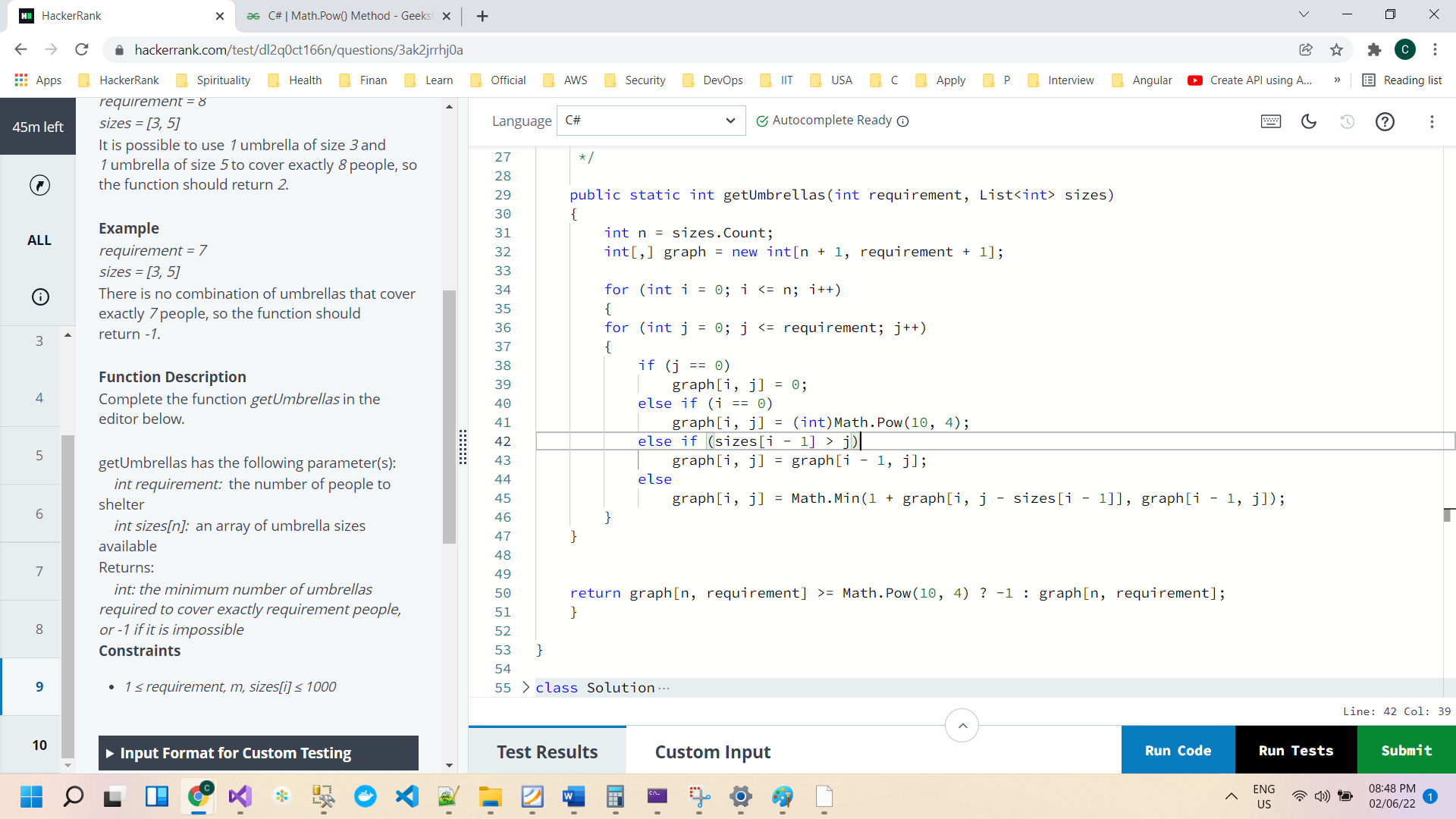
                finalResult += i;

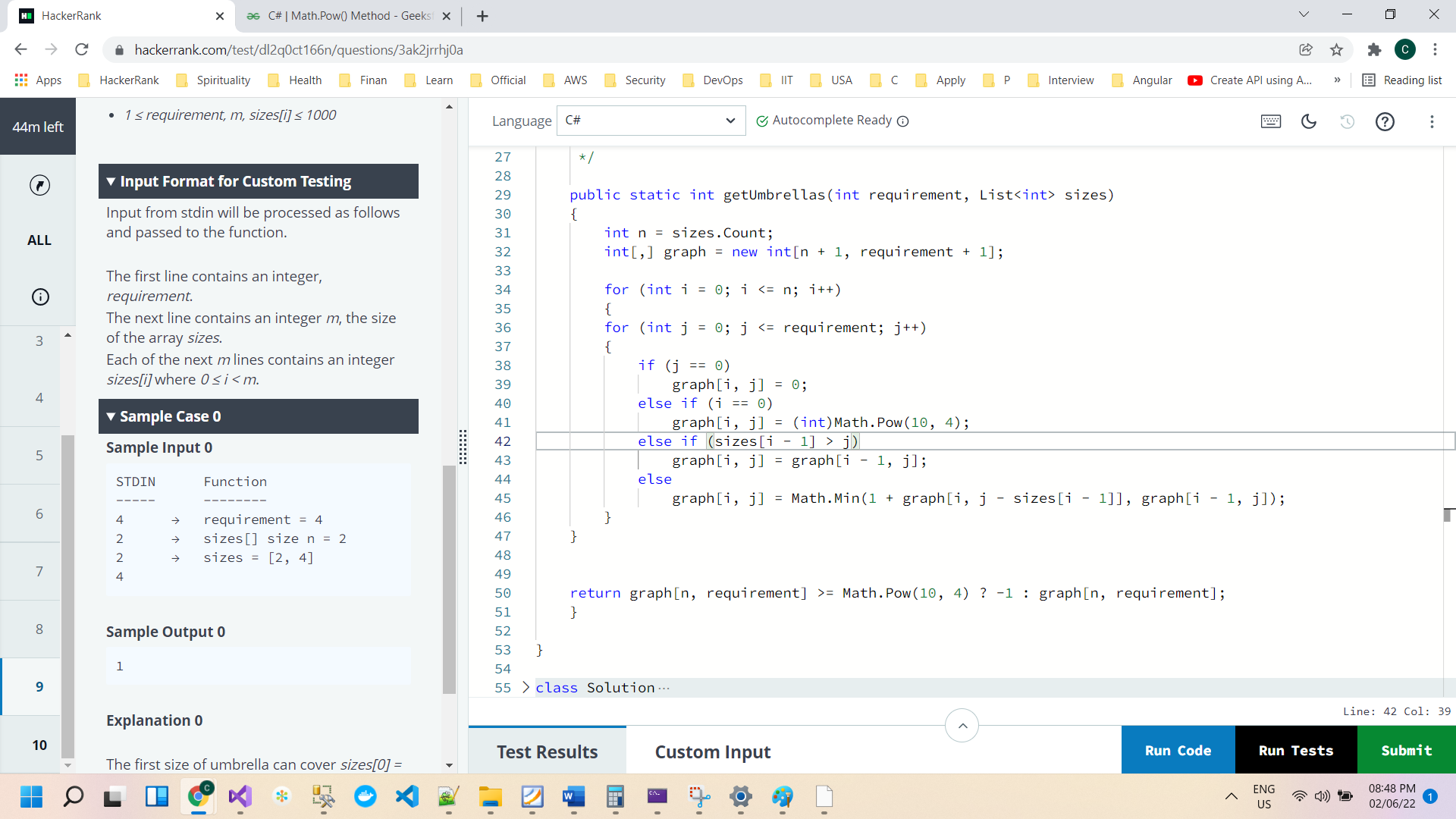
            }

            return finalResult;

    }







public static int getUmbrellas(int requirement, List<int> sizes)

{

int n = sizes.Count;

int[,] graph = new int[n + 1, requirement + 1];

for (int i = 0; i <= n; i++)

{

for (int j = 0; j <= requirement; j++)

{

if (j == 0)

graph[i, j] = 0;

else if (i == 0)

graph[i, j] = (int)Math.Pow(10, 4);

else if (sizes[i - 1] > j)

graph[i, j] = graph[i - 1, j];

else

graph[i, j] = Math.Min(1 + graph[i, j - sizes[i - 1]], graph[i - 1, j]);

}

}

return graph[n, requirement] >= Math.Pow(10, 4) ? -1 : graph[n, requirement];

}

Test case

Console.WriteLine(getUmbrellas(8, new List<int>() { 3, 5 }));

Console.WriteLine(getUmbrellas(11, new List<int>() { 1, 5, 6, 9 }));

Console.WriteLine(getUmbrellas(8, new List<int>() { 5, 4, 4, 2 }));

Console.WriteLine(getUmbrellas(7, new List<int>() { 3, 5 }));

Console.WriteLine(getUmbrellas(5, new List<int>() { 3, 5 }));

Console.WriteLine(getUmbrellas(755, new List<int>() { 151, 6, 19, 46, 27, 26, 25, 42, 20, 17, 15, 45, 5, 20, 3, 1, 48, 46, 43, 5, 18, 16, 48, 48, 34, 48, 25, 29, 25, 32, 5, 23, 5, 15, 31, 17, 28, 34, 11, 38, 48, 40, 40, 40, 6, 5, 47, 25, 49, 3, 50, 28, 3, 23, 37, 45, 28, 18, 36, 6, 49, 8, 35, 39, 42, 31, 44, 6, 42, 5, 22, 36, 12, 4, 20, 42, 45, 36, 8, 5, 26, 5, 12, 50, 30, 19, 44, 19, 45, 41, 12, 48, 46, 50, 30, 38, 18, 19, 36, 5, 25, 39, 19, 28, 36, 22, 13, 46, 17, 6, 22, 25, 13, 1, 21, 24, 29, 3, 38, 6, 39, 6, 42, 33, 38, 38, 35, 30, 12, 49, 21, 19, 24, 15, 5, 44, 27, 12, 22, 49, 41, 1, 49, 49, 28, 3, 17, 45, 3, 27, 47, 50, 46, 4, 13, 28, 35, 49, 4, 27, 9, 32, 11, 35, 15, 23, 50, 32, 35, 30, 20, 46, 37, 3, 46, 15, 48, 48, 3, 45, 20, 23, 6, 32, 17, 14, 9, 10, 33, 24, 20, 18, 12, 30, 14, 4, 19, 49, 13, 50, 23, 35, 2, 27, 14, 16, 21, 41, 31, 35, 14, 9, 7, 46, 4, 42, 48, 48, 21, 37, 30, 31, 46, 21, 5, 47, 44, 44, 28, 3, 9, 2, 21, 19, 39, 41, 25, 50, 50 }));