Q1 Write an efficient algorithm that searches for a value target in an m x n integer matrix. This matrix has the following properties:

1. Integers in each row are sorted from right to left.
2. The first integer of each row is greater than the last integer of the previous row.

Example-:

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

Output: True

```
def search(matrix, t):
    if not matrix or not matrix[0]:
        return False

    rows, cols = len(matrix), len(matrix[0])
    left, right = 0, rows * cols - 1

    while left <= right:
        mid = (left + right) // 2
        mid_value = matrix[mid // cols][mid % cols]

        if mid_value == t:
            return True
        elif mid_value < t:
            left = mid + 1
        else:
            right = mid - 1

    return False
matrix = [[1, 3, 5, 7], [10, 11, 16, 20], [23, 30, 34, 60]]
t = 3
output = search(matrix, t)
print(output)
```

Q2. **2.** Write a program that takes a string as input, and counts the frequency of each word in the string, there might be repeated characters in the string. Your task is to find the highest frequency and returns the length of the highest-frequency word.

Note - You have to write at least 2 additional test cases in which your program will run successfully and provide an explanation for the same.

Example input - string = “write write write all the number from from from 1 to 100”

Example output - 5

Explanation - From the given string we can note that the most frequent words are “write” and “from” and the maximum value of both the values is “write” and its corresponding length is 5

```
def highest_frequency(int _str ):
    word_count = {}
    words = int _str .split()
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1
    max_word = max(word_count, key=word_count.get)
    return len(max_word)
int _str = "write all the number from from from 1 to 100"
output = highest_frequency(int _str )
print(output)
```