**COURSE CODE : CSA0836**

**COURSE : PYTHON PROGRAMMING**

**NAME : KAVALI SAI KUMAR**

**REGISTER NUMBER : 192110215**

**DAY 5 PROGRAMS**

1. Given a string s consisting of words and spaces, return the length of the last word in the string.A word is a maximal substring consisting of non-space characters only.

Test Case:

Input: s = "Hello World"

Output: 5

Input: s = " fly me to the moon "

Output: 4

Input: s = "luffy is still joyboy"

Output: 6

Input: s = "123"

Output: 3

Input: s = " 45&29 8\*6^4"

Output: 5

**def length(str):**

**lis = list(str.split(" "))**

**return len(lis[-1])**

**str = "fly me to the moon"**

**print("The length of last word is",**

**length(str))**

1. Given an integer n, return the least number of perfect square numbers that sum to n. A perfect square is an integer that is the square of an integer; in other words, it is the product of some integer with itself. For example, 1, 4, 9, and 16 are perfect squares while 3 and 11 are not.

Test cases:

1. Input: n = 12, Output: 3

2. Input: n = 13, Output: 2

3. Input: n= 1, Output: 1

4. Input: n=4, Output: 2

5. Input: n=3, Output: 1

**def minJumps(arr, l, h):**

**if (h == l):**

**return 0**

**if (arr[l] == 0):**

**return float('inf')**

**min = float('inf')**

**for i in range(l + 1, h + 1):**

**if (i < l + arr[l] + 1):**

**jumps = minJumps(arr, i, h)**

**if (jumps != float('inf') and**

**jumps + 1 < min):**

**min = jumps + 1**

**return min**

**arr=eval(input("Enter list:"))**

**n=len(arr)**

**print('Minimum number of jumps to reach',**

**'end is', minJumps(arr, 0, n-1))**

1. Given a matrix of size N \* N containing only 0s and 1s, where 0 represents white and 1 represents black. The task is to minimize the number of swaps to form a valid chessboard. Only 2 rows or 2 columns can be swapped with each other.If it is impossible to form a chessboard, return -1.

Test Case:

1.Input: board = [[0,1,1,0],[0,1,1,0],[1,0,0,1],[1,0,0,1]] Output: 2

2.Input: board = [[0,1],[1,0]] Output: 0

3.Input: board = [[1,0],[1,0]] Output: -1

4.Input: [[0, 1, 0], [1, 0, 1], [1, 1, 0]] Output: -1

5.Input: [[0, 1, 1, 0], [0, 1, 1, 0], [1, 0, 0, 1], [1, 0, 0, 1]] Output: 2

**m1 = [[5,2,7],**

**[3,1,5]]**

**m2 = [[4,8,9],**

**[1,3,1]]**

**res = [[0, 0, 0],**

**[0, 0, 0],**

**[0, 0, 0]]**

**for i in range(len(m1)):**

**for j in range(len(m2[0])):**

**for k in range(len(m2)):**

**res[i][j] += m1[i][k] \* m2[k][j]**

**for r in res:**

**print(r)**

1. Reverse Words in a String Given an input string s, reverse the order of the words. A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space. Return a string of the words in reverse order concatenated by a single space. Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

1.Input: s = "the sky is blue"

Output: "blue is sky the"

2.Input: s = " hello world "

Output: "world hello"

3.Input: s = "a good example"

Output: "example good a"

4. s= “apple is red”

5.s= “Red rose”

**s = "i like this program very much"**

**words = s.split(' ')**

**string =[]**

**for word in words:**

**string.insert(0, word)**

**print("Reversed String:")**

**print(" ".join(string))**

1. Longest Substring with At Least K Repeating Characters Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k. s consists of only lowercase English letters.

Test cases:

1.Input: s = "aaabb", k = 3

Output: 3

2.Input: s = "ababbc", k = 2

Output: 5

3. s= “ababab”, k=3

4. s=”xyzxyz” k =2

5.s=12345, k=0

**def longestSubstring(s, k) :**

**ans = 0**

**freq = [0]\*26**

**n = len(s)**

**for i in range(n) :**

**freq[ord(s[i]) - ord('a')] += 1**

**unique = 0**

**for i in range(26) :**

**if (freq[i] != 0) :**

**unique += 1**

**for curr\_unique in range(1, unique + 1) :**

**Freq = [0]\*26**

**start, end = 0, 0**

**cnt, count\_k = 0, 0**

**while (end < n) :**

**if (cnt <= curr\_unique) :**

**ind = ord(s[end]) - ord('a')**

**if (Freq[ind] == 0) :**

**cnt += 1**

**Freq[ind] += 1**

**if (Freq[ind] == k) :**

**count\_k += 1**

**end += 1**

**else :**

**ind = ord(s[start]) - ord('a')**

**if (Freq[ind] == k) :**

**count\_k -= 1**

**Freq[ind] -= 1**

**if (Freq[ind] == 0) :**

**cnt -= 1**

**start += 1**

**if ((cnt == curr\_unique) and (count\_k == curr\_unique)) :**

**ans = max(ans, end - start)**

**print(ans)**

**S = "aaabb"**

**K = 3**

**longestSubstring(S, K)**