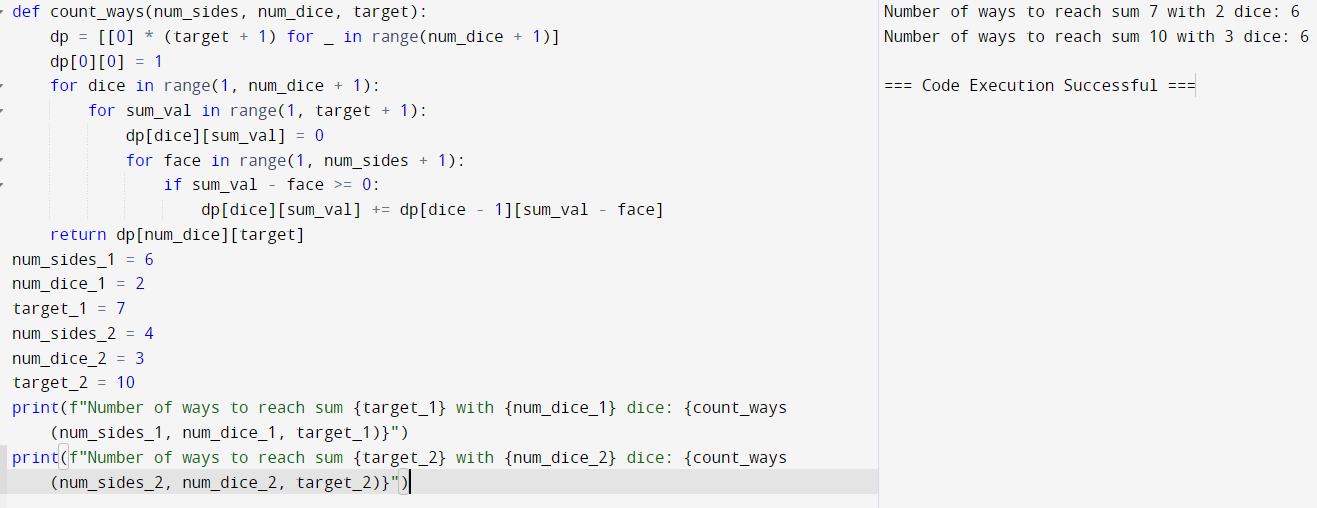
1. You are given the number of sides on a die (num\_sides), the number of dice to throw

(num\_dice), and a target sum (target). Develop a program that utilizes dynamic

programming to solve the Dice Throw Problem.



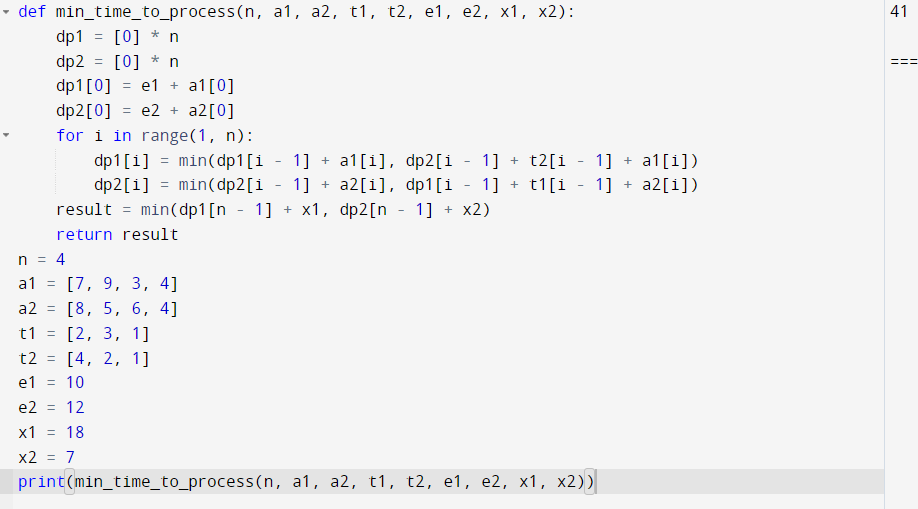
2. In a factory, there are two assembly lines, each with n stations. Each station performs a

specific task and takes a certain amount of time to complete. The task must go through each

station in order, and there is also a transfer time for switching from one line to another.

Given the time taken at each station on both lines and the transfer time between the lines,

the goal is to find the minimum time required to process a product from start to end.



3. An automotive company has three assembly lines (Line 1, Line 2, Line 3) to produce

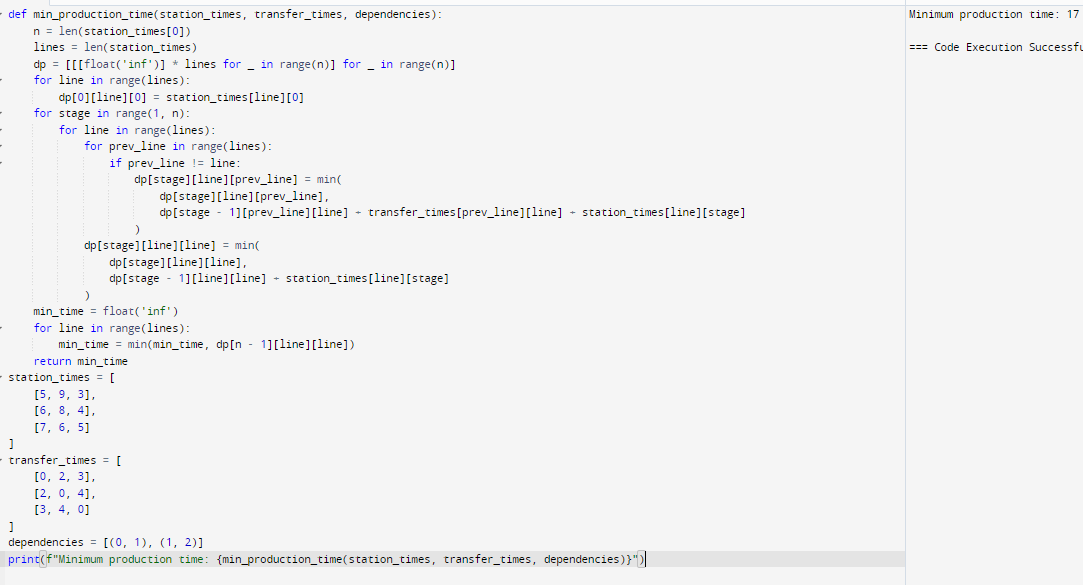
different car models. Each line has a series of stations, and each station takes a certain

amount of time to complete its task. Additionally, there are transfer times between lines,

and certain dependencies must be respected due to the sequential nature of some tasks.

Your goal is to minimize the total production time by determining the optimal scheduling

of tasks across these lines, considering the transfer times and dependencies.



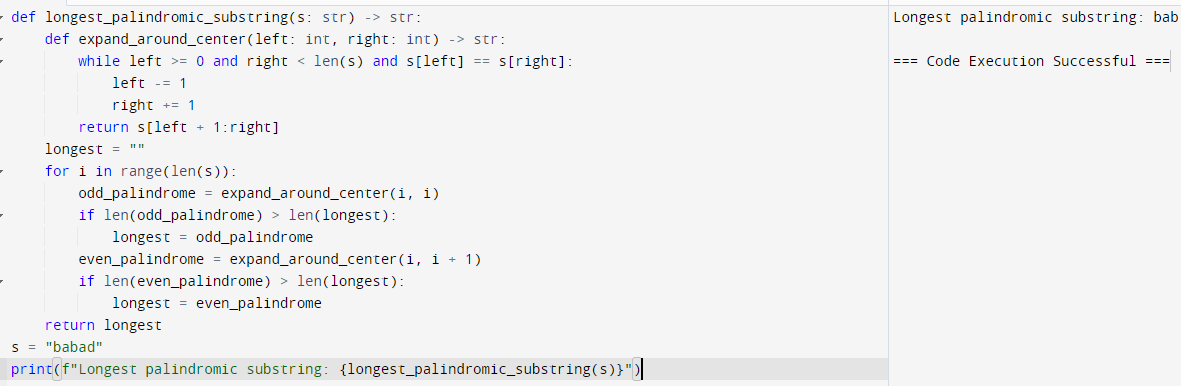
4. Write a c program to find the minimum path distance by using matrix form.



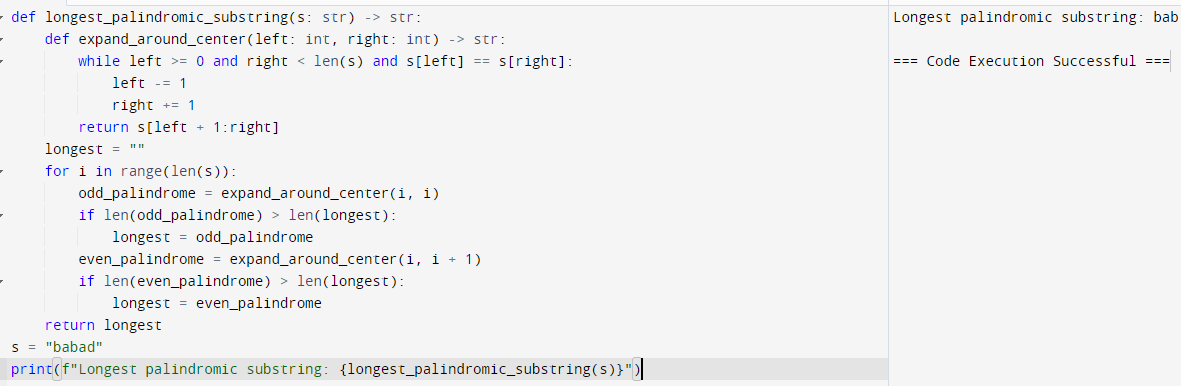
5. Assume you are solving the Traveling Salesperson Problem for 4 cities (A, B, C, D) with

known distances between each pair of cities. Now, you need to add a fifth city (E) to the

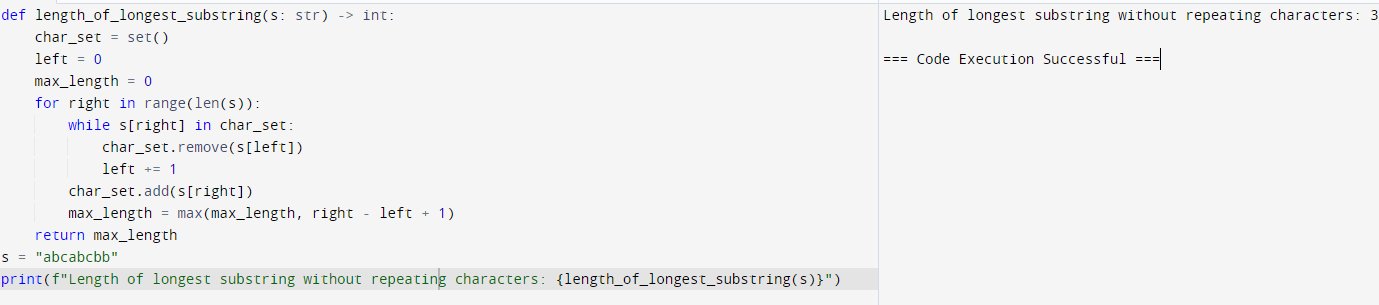
problem.



6. Given a string s, return the longest palindromic substring in S.



7. Given a string s, find the length of the longest substring without repeating characters.

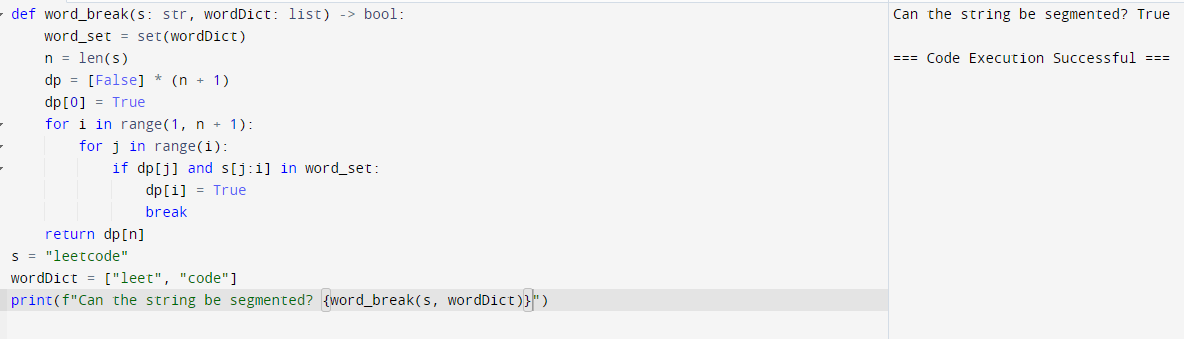


8. Given a string s and a dictionary of strings wordDict, return true if s can be segmented into

a space-separated sequence of one or more dictionary words.

Note that the same word in the dictionary may be reused multiple times in the

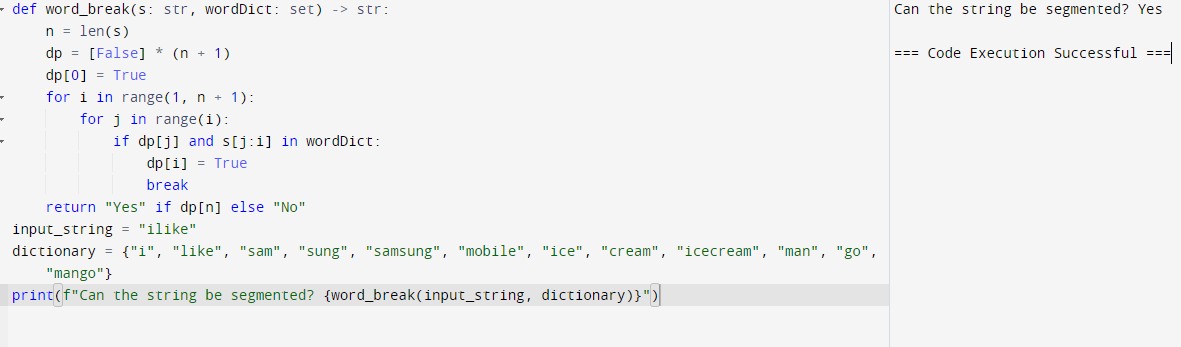
segmentation.



9. Given an input string and a dictionary of words, find out if the input string can be segmented

into a space-separated sequence of dictionary words.Consider the following dictionary { i,

like, sam, sung, samsung, mobile, ice, cream, icecream, man, go, mango}



10. Given an array of strings words and a width maxWidth, format the text such that each line

has exactly maxWidth characters and is fully (left and right) justified. You should pack your

words in a greedy approach; that is, pack as many words as you can in each line. Pad extra

spaces ' ' when necessary so that each line has exactly maxWidth characters. Extra spaces

between words should be distributed as evenly as possible. If the number of spaces on a line

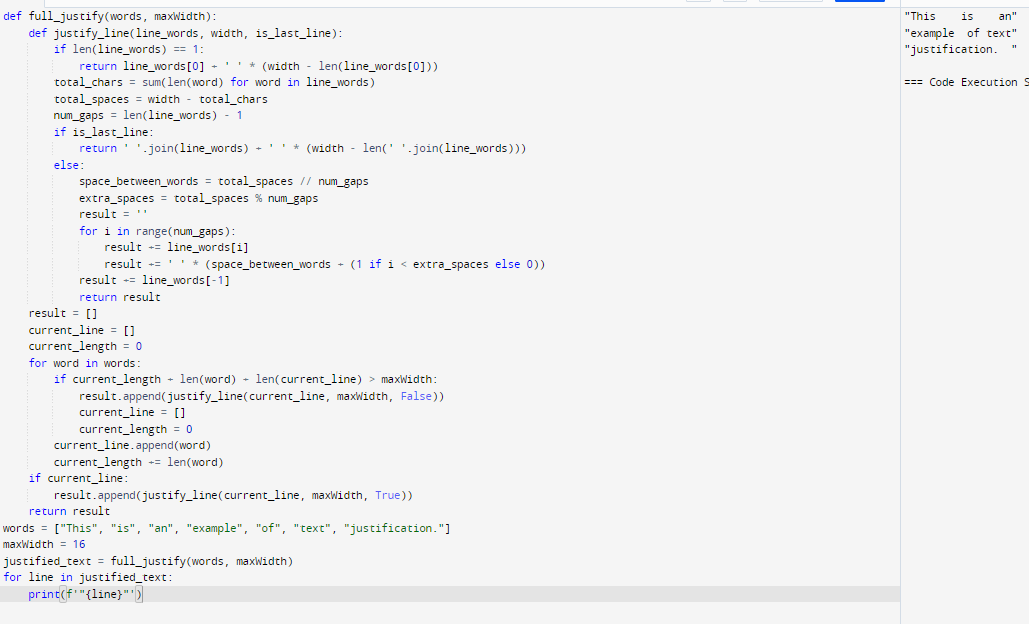
does not divide evenly between words, the empty slots on the left will be assigned more

spaces than the slots on the right. For the last line of text, it should be left-justified, and no

extra space is inserted between words. A word is defined as a character sequence consisting

of non-space characters only. Each word's length is guaranteed to be greater than 0 and not

exceed maxWidth. The input array words contains at least one word.



11. Design a special dictionary that searches the words in it by a prefix and a suffix. Implement

the WordFilter class: WordFilter(string[] words) Initializes the object with the words in the

dictionary.f(string pref, string suff) Returns the index of the word in the dictionary, which

has the prefix pref and the suffix suff. If there is more than one valid index, return the

largest of them. If there is no such word in the dictionary, return -1.

