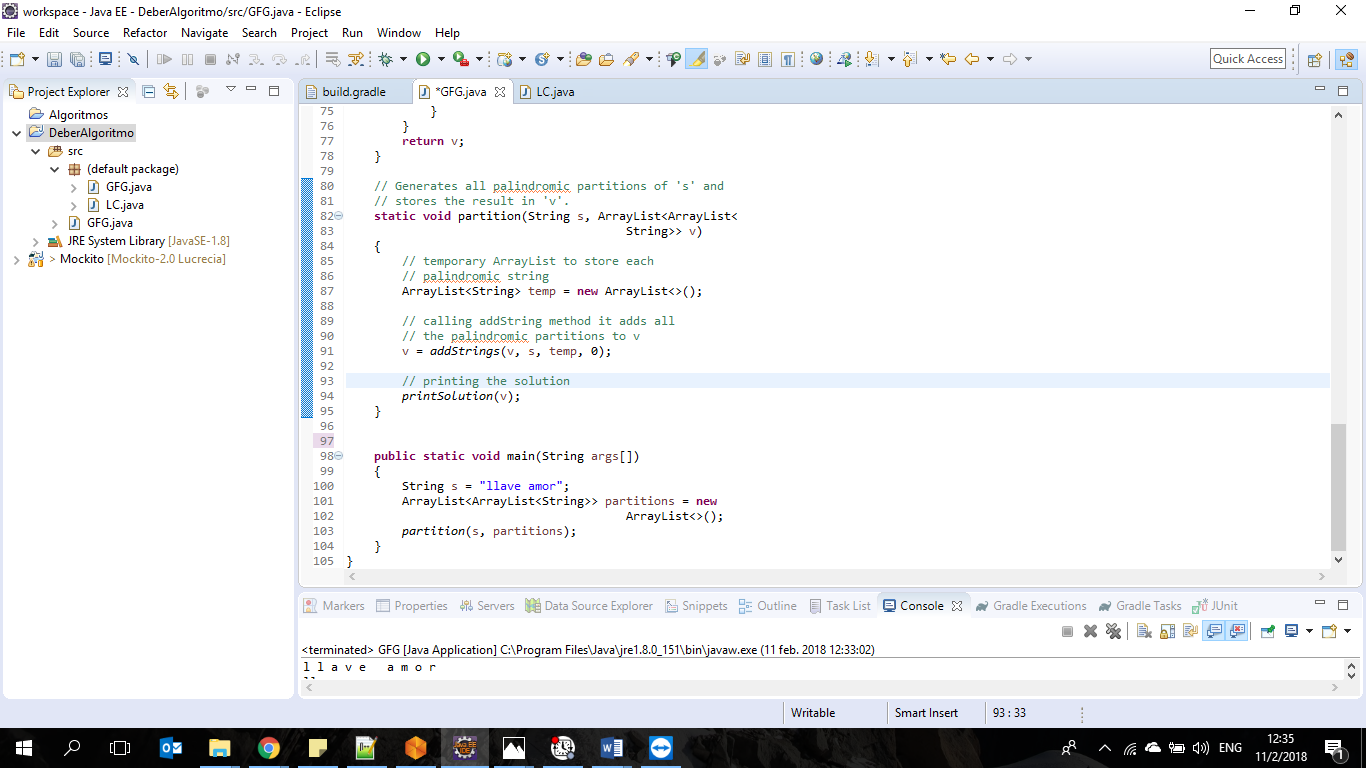
1. Print all palindromic partitions of a string

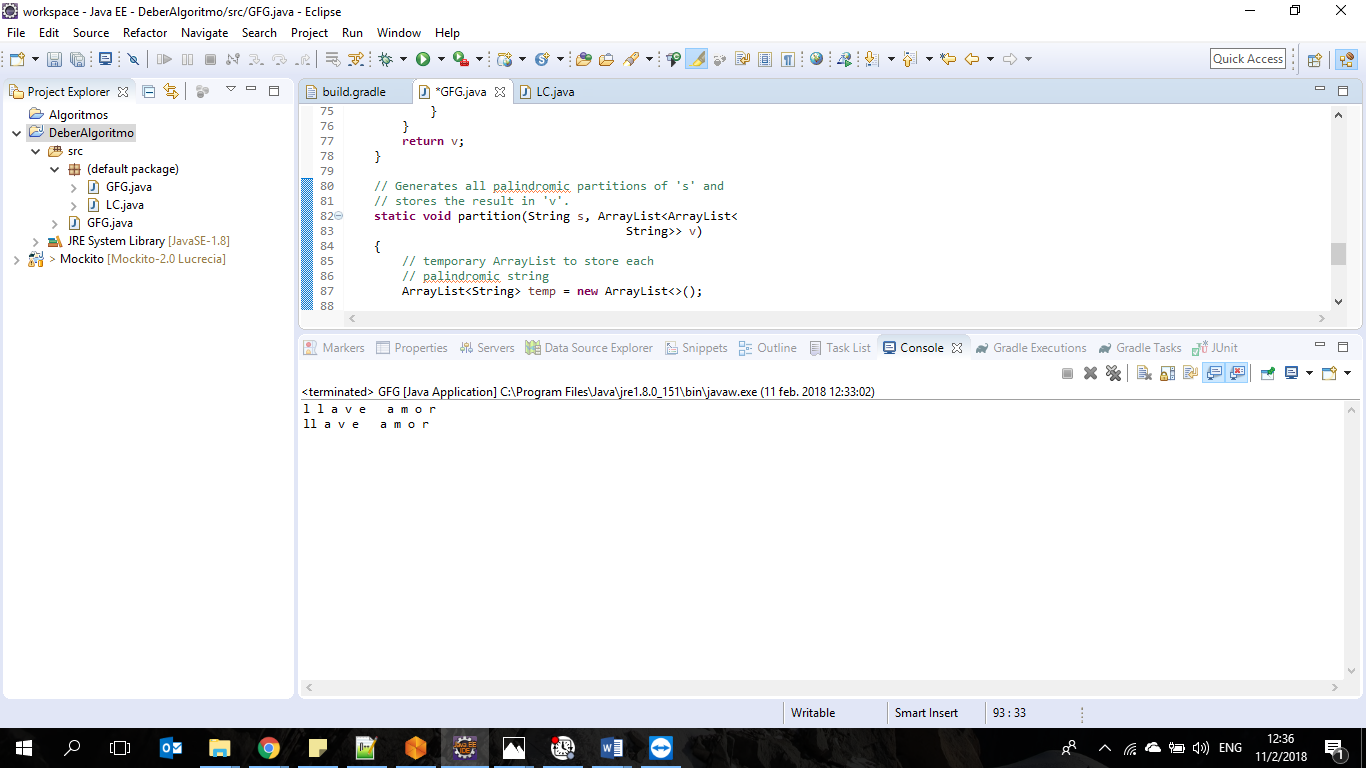
Given a string s, partition s such that every string of the partition is a palindrome. Return all possible palindrome partitioning of s.

CODIGO FUENTE EN JAVA



RESULTADO

Palabras: LLAVE, AMOR



CODIGO FUENTE

// Java program to print all palindromic partitions

// of a given string.

import java.util.ArrayList;

public class GFG

{

// Returns true if str is palindrome, else false

static boolean checkPalindrome(String str)

{

int len = str.length();

len--;

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

{

if (str.charAt(i) != str.charAt(len))

return false;

len--;

}

return true;

}

// Prints the partition list

static void printSolution(ArrayList<ArrayList<String>>partitions)

{

for(ArrayList<String> i: partitions)

{

for(String j: i)

{

System.out.print(j+" ");

}

System.out.println();

}

}

static ArrayList<ArrayList<String>> addStrings(ArrayList<

ArrayList<String>> v, String s, ArrayList<String> temp, int index)

{

int len = s.length();

String str = "";

ArrayList<String> current = new ArrayList<>(temp);

if (index == 0)

temp.clear();

// Iterate over the string

for (int i = index; i < len; ++i)

{

str = str + s.charAt(i);

// check whether the substring is

// palindromic or not

if (checkPalindrome(str))

{

// if palindrome add it to temp list

temp.add(str);

if (i + 1 < len)

{

// recurr to get all the palindromic

// partitions for the substrings

v = addStrings(v,s,temp,i+1);

}

else

{

// if end of the string is reached

// add temp to v

v.add(temp);

}

// temp is reinitialize with the

// current i.

temp = new ArrayList<>(current);

}

}

return v;

}

// Generates all palindromic partitions of 's' and

// stores the result in 'v'.

static void partition(String s, ArrayList<ArrayList<

String>> v)

{

// temporary ArrayList to store each

// palindromic string

ArrayList<String> temp = new ArrayList<>();

// calling addString method it adds all

// the palindromic partitions to v

v = addStrings(v, s, temp, 0);

// printing the solution

printSolution(v);

}

public static void main(String args[])

{

String s = "llave amor";

ArrayList<ArrayList<String>> partitions = new

ArrayList<>();

partition(s, partitions);

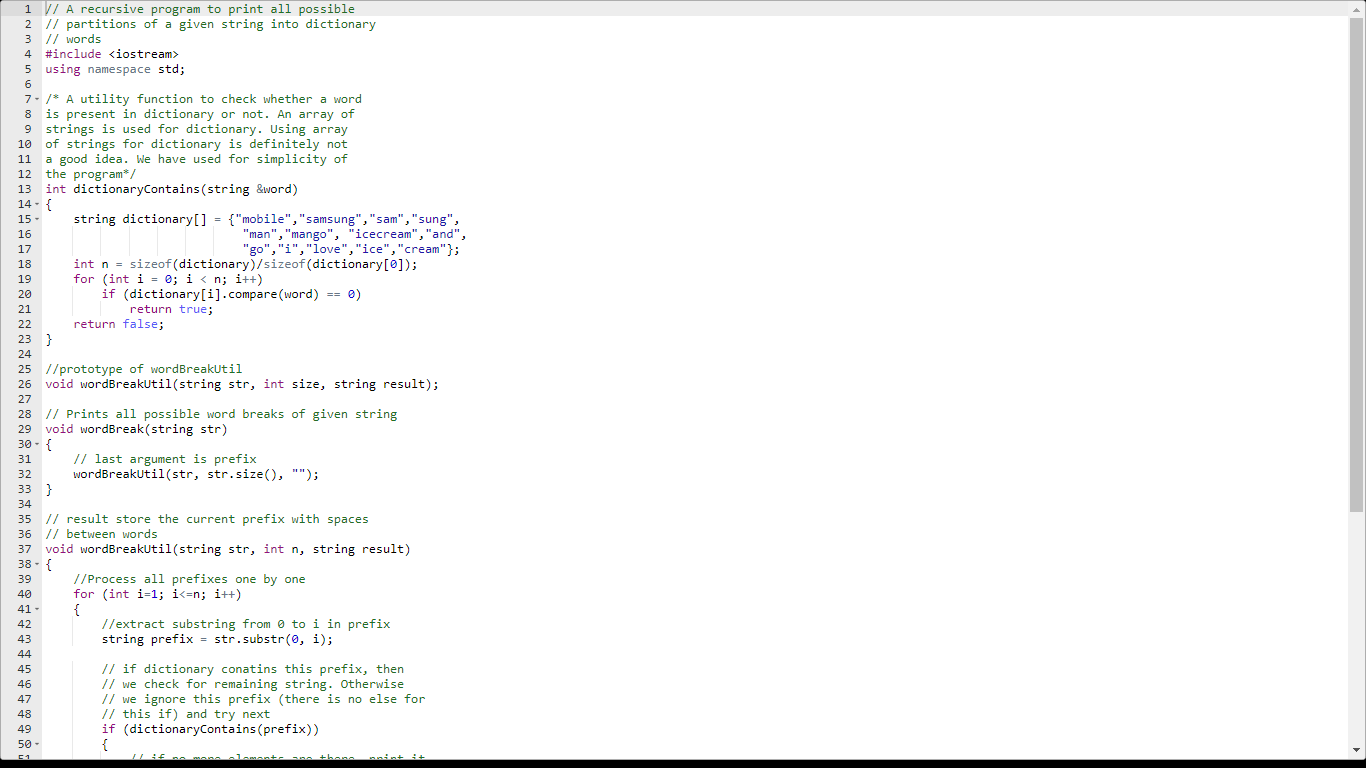
}

}

# 2) Word Break Problem using Backtracking

Given a valid sentence without any spaces between the words and a dictionary of valid English words, find all possible ways to break the sentence in individual dictionary words.

CODIGO FUENTE EN C



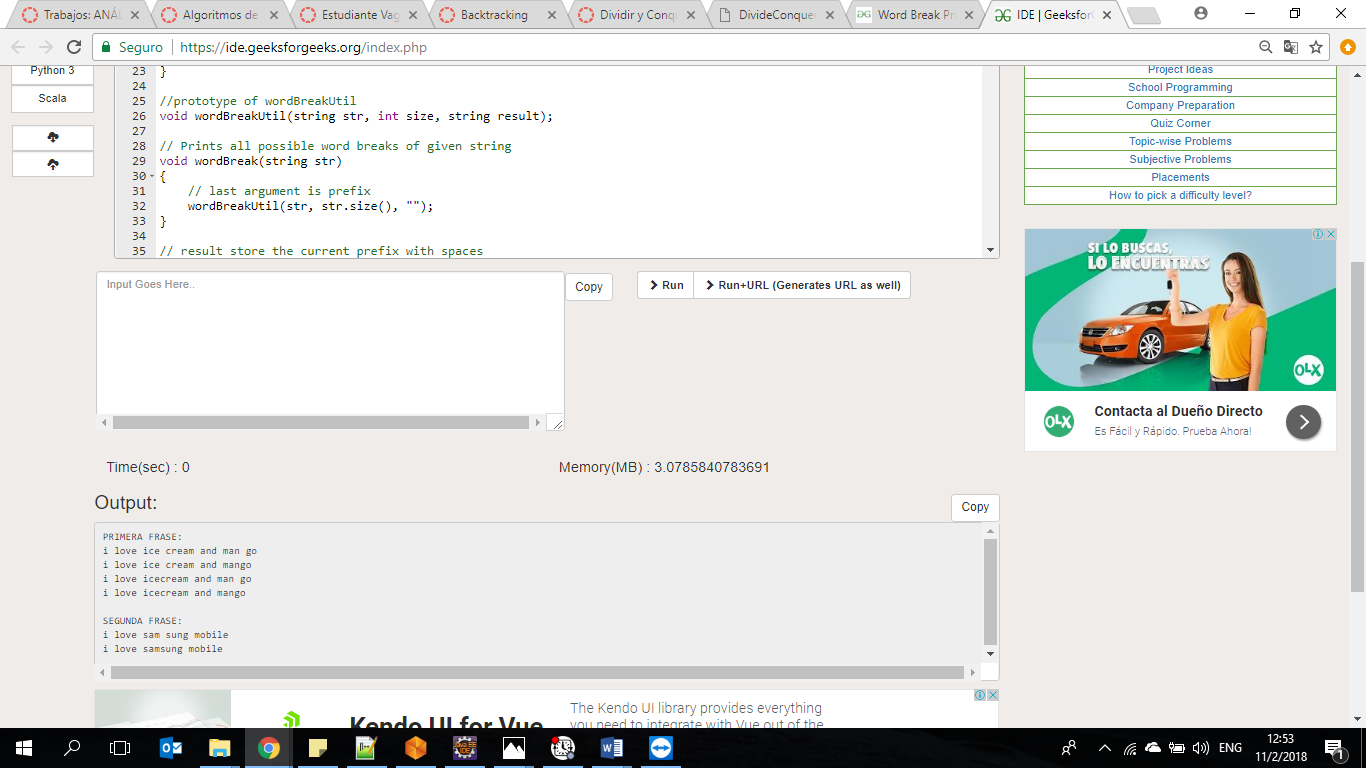
RESULTADO

Palabras Clave:



Primera Frase: iloveicecreamandmango

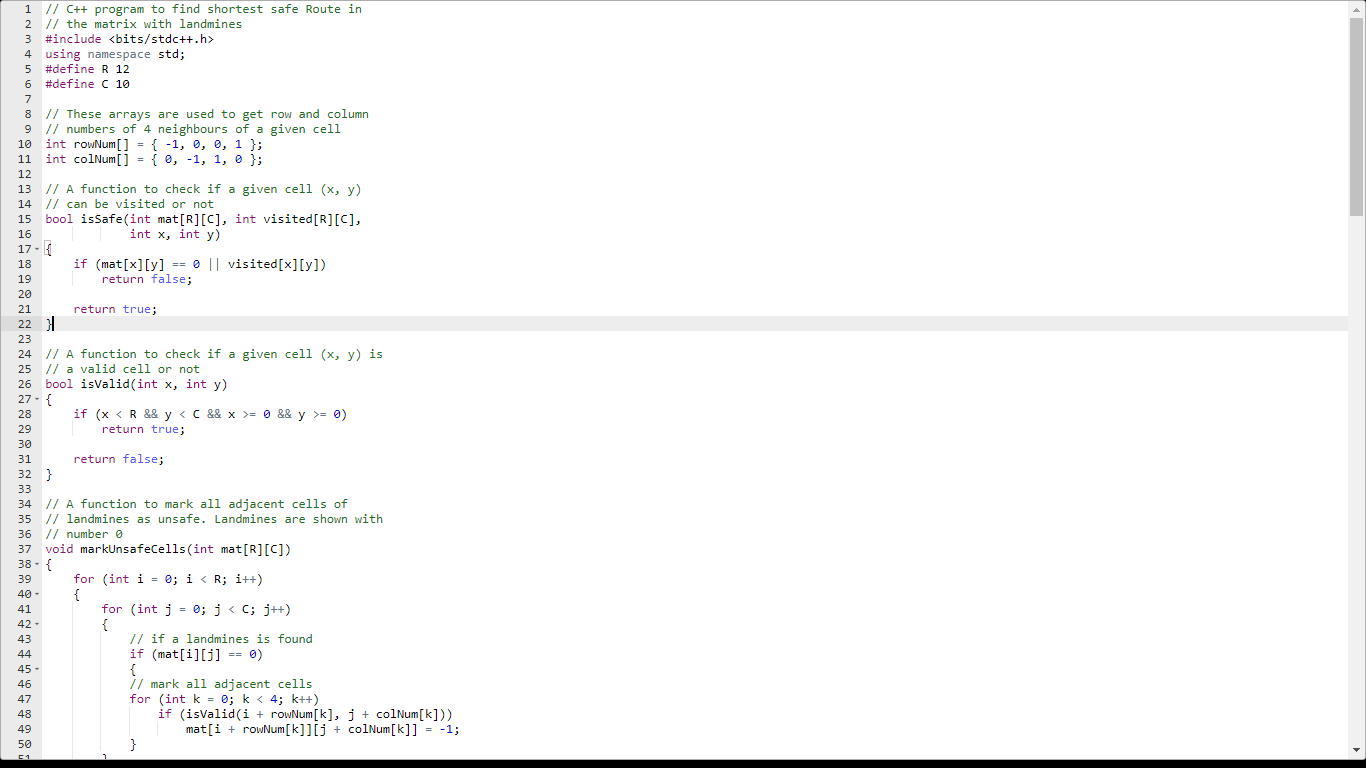
Segunda Frase: ilovesamsungmobile



# 3) Find shortest safe route in a path with landmines

Given a path in the form of a rectangular matrix having few landmines arbitrarily placed (marked as 0), calculate length of the shortest safe route possible from any cell in the first column to any cell in the last column of the matrix. We have to avoid landmines and their four adjacent cells (left, right, above and below) as they are also unsafe. We are allowed to move to only adjacent cells which are not landmines. i.e. the route cannot contains any diagonal moves.

CODIGO FUENTE



RESULTADO

