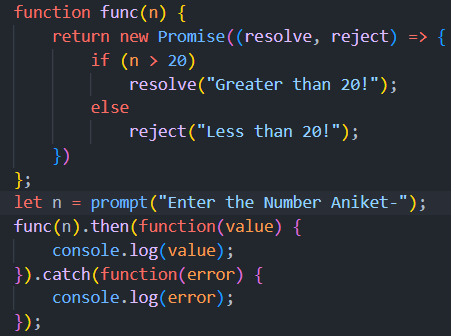
# **ASSIGNMENT 7**

ANIKET RAY

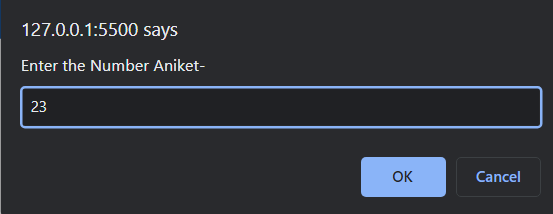
1NT19IS027

Q1) **Write a function numTest that takes a number as an argument and returns a Promise that tests if the value is less than or greater than the value 20.**

CODE

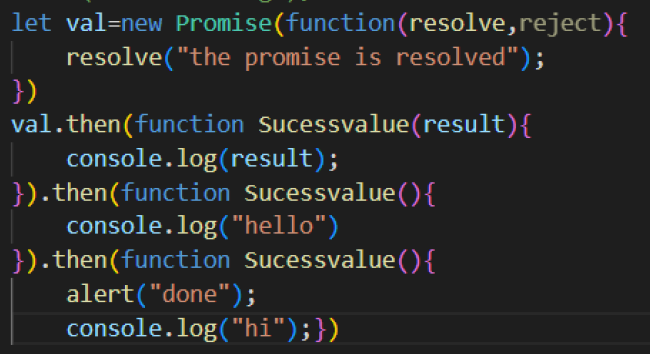


OUTPUT

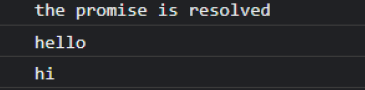
 

Q2) **Write a JavaScript code to handle multiple call back functions using JavaScript promises (use promiseobject.then(onfulfilled,onrejected)).**

CODE

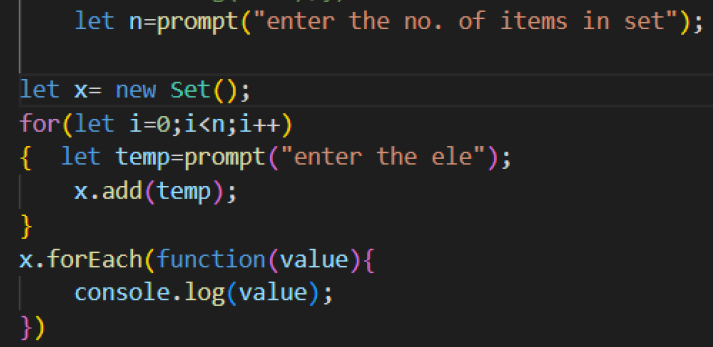


OUTPUT

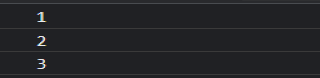


Q3) **Write a program to store values into a set, and to retrieve value from the set, to iterate over the set**

CODE

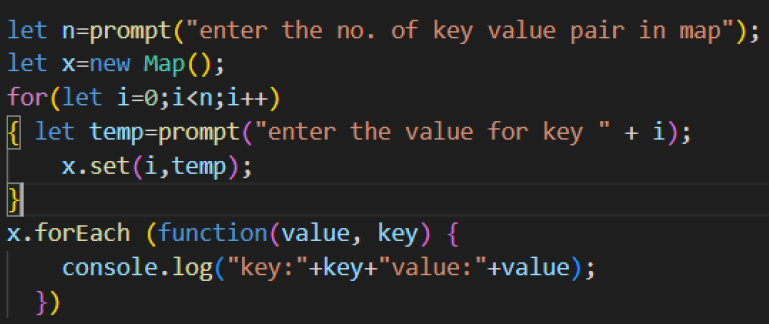


OUTPUT

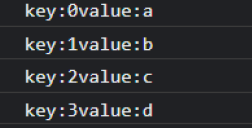


Q4) **Write a program to store values into a map, and to retrieve value from the map using key, to iterate over the map**

CODE

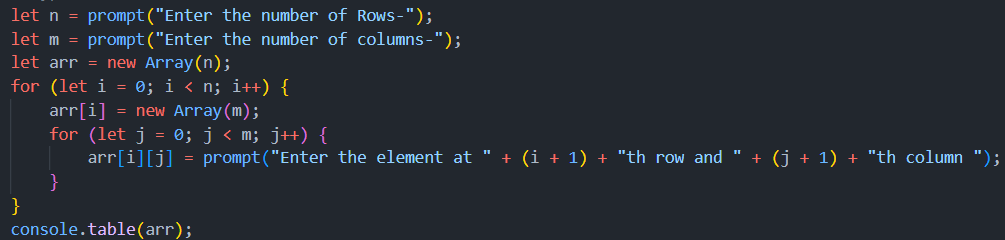


OUTPUT

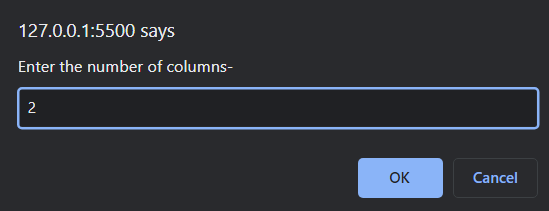
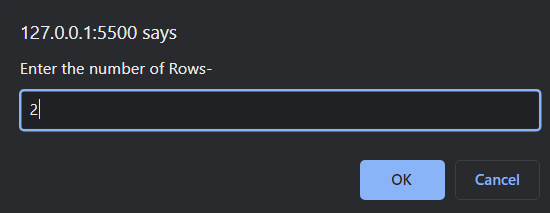


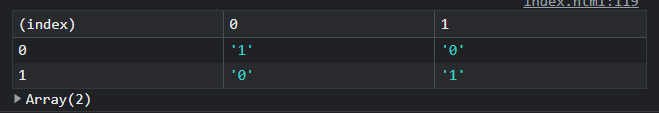
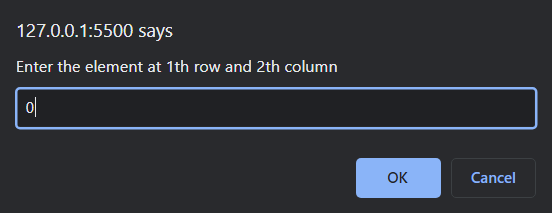
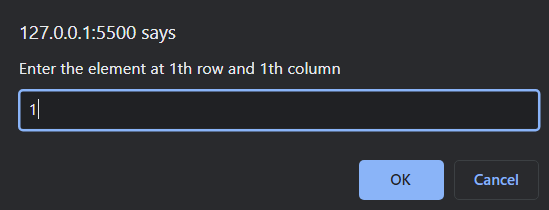
Q5) **Write a program to iterate over a 2-dimensional array and print all the values of it**

CODE



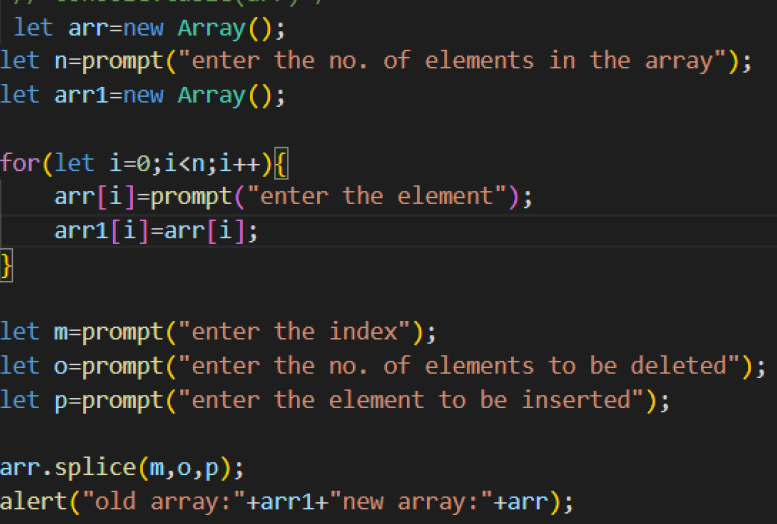
OUTPUT



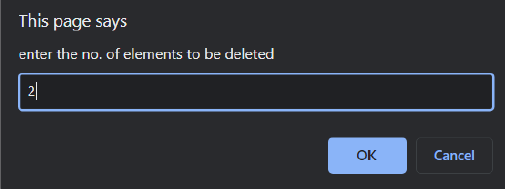
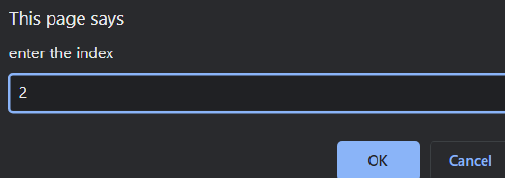
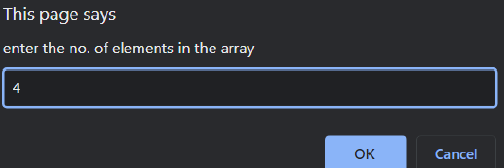


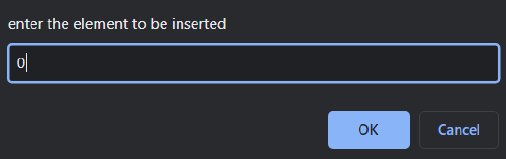
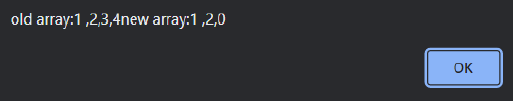
Q6) **Write a JavaScript code to insert and remove elements from the array for the given index**.

CODE



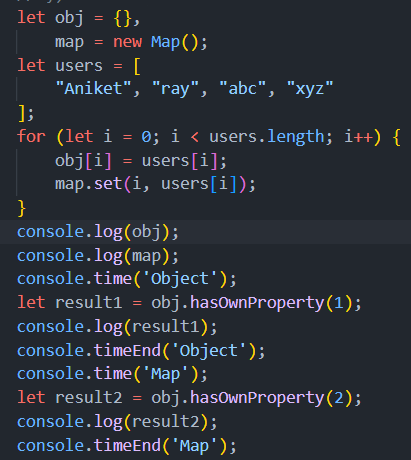
OUTPUT



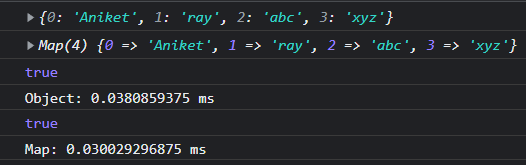
 

Q7) **Show how map is different from object to store key value pairs with coding example and prove Maps perform better than objects in most of the scenarios involving addition and removal of keys**.

CODE

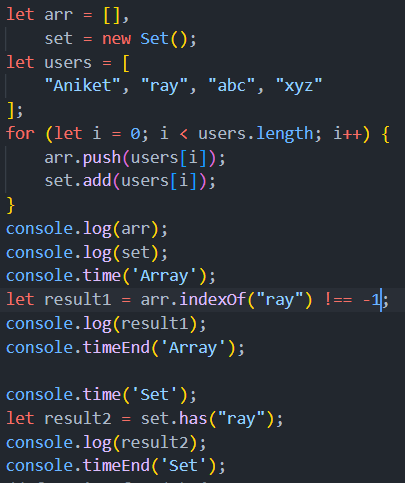


OUTPUT

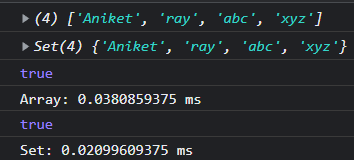


Q8) **Show how set is different from array to store the value with coding example and prove Sets perform better than Arrays in most of the scenarios involving searching values present in it.**

CODE

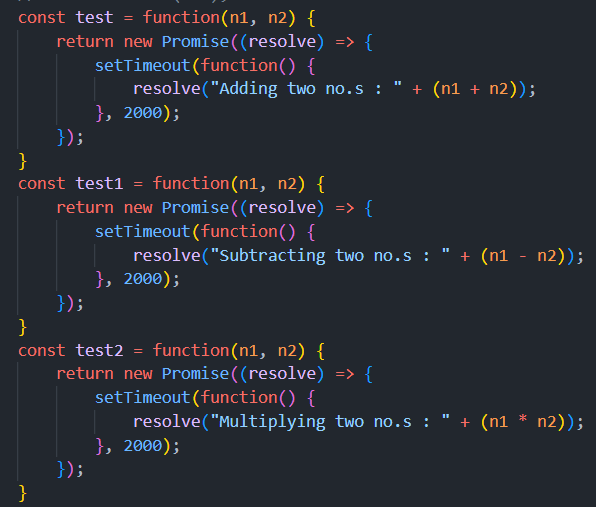
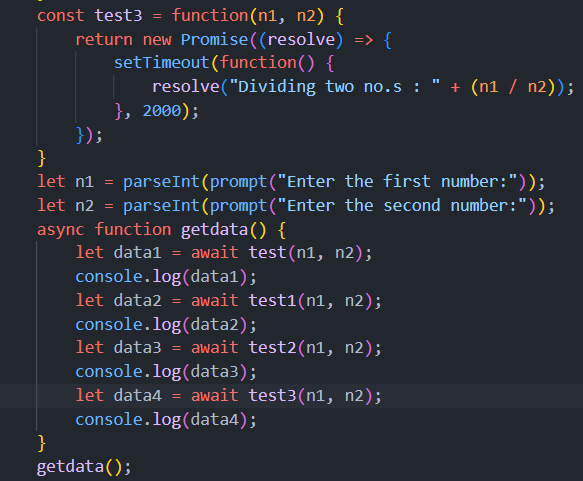


OUTPUT

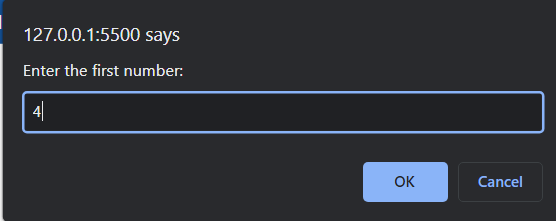
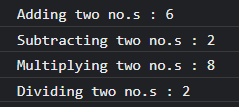
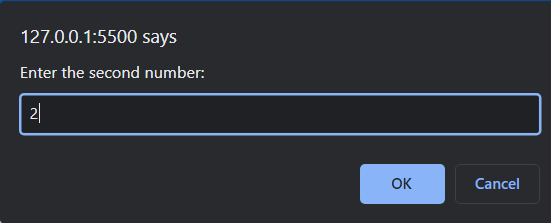


Q9)**Implement a JavaScript promise to perform arithmetic operations. Display result for each operation synchronously using await () method. (Give delay in each promise object using settimeout() method**

CODE

OUTPUT

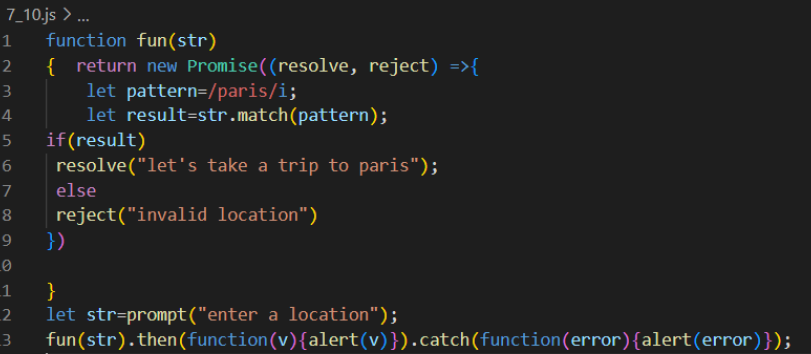
 

Q10)**Write a Javascript program where user passes the location and a function is called which returns a promise, if the location passed is Paris Below is the output expected:**

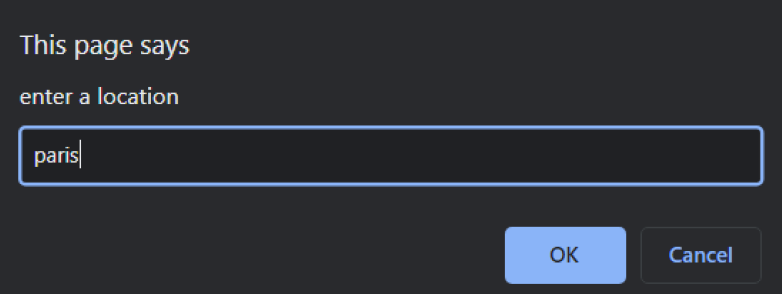
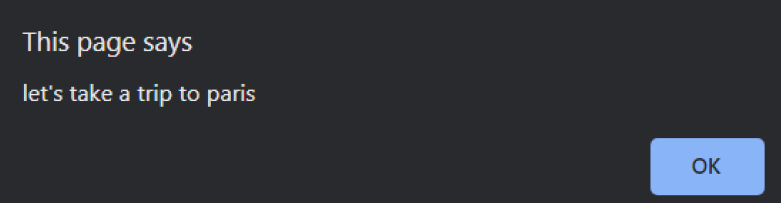
**"Let's take a trip to Paris"**

**If the location is other than Paris, show the error message "Invalid Location"**

**CODE**



**OUTPUT**

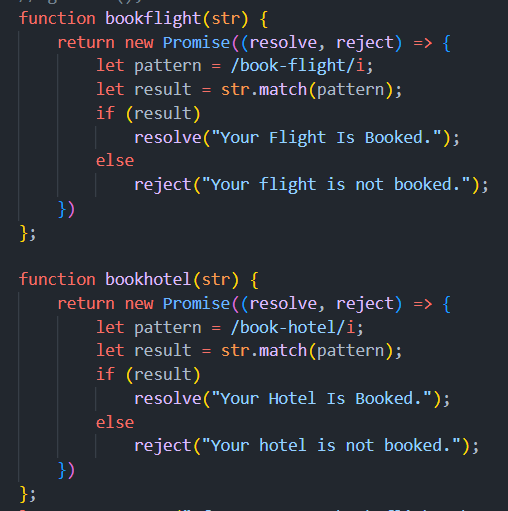
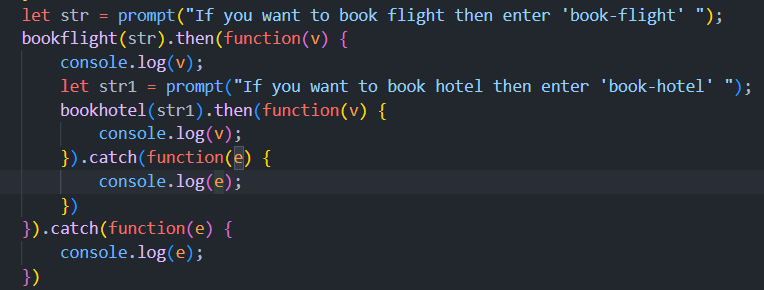
**** 

Q11) **Write a JavaScript program to book a hotel only after booking a flight.**

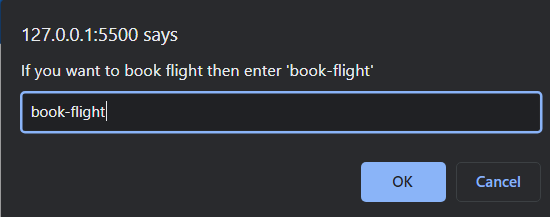
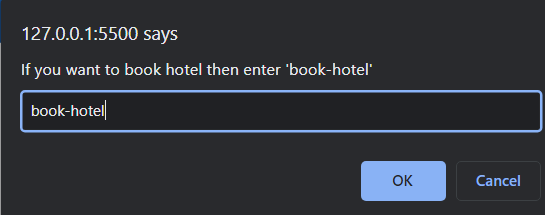
**[Hint: To achieve this, the promise returned from the bookHotel function is resolved only after resolving the promise from bookFlight function.**

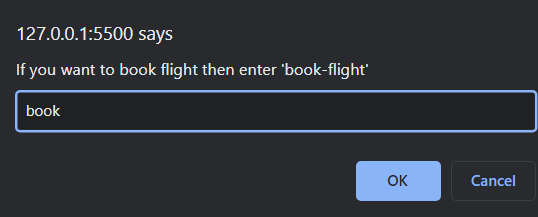
**If the promise gets rejected from bookflight then it won't execute the second function.]**

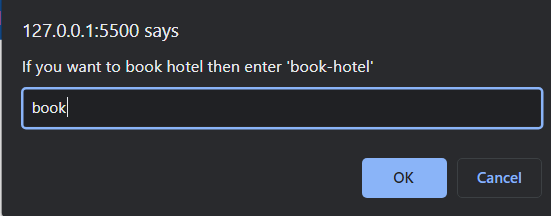
**CODE**

** **

**OUTPUT**

Q12) **Write a program that prints two numbers: the numbers of cows and**

**chickens on a farm, with the words Cows and Chickens after them and**

**zeros padded before both numbers so that they are always three digits**

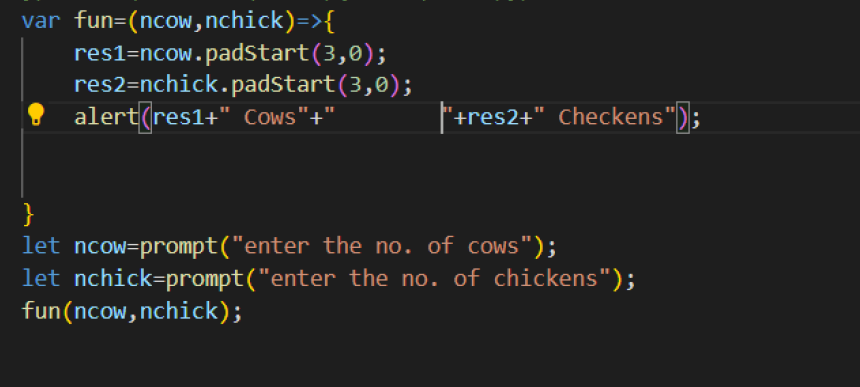
**long using functions**

**Input (Function Call): printFarmInventory(7, 11);**

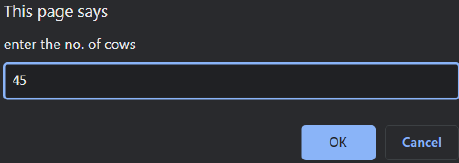
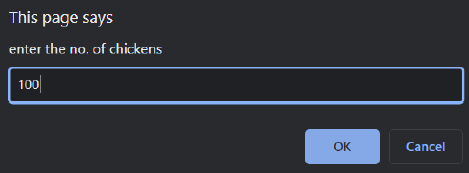
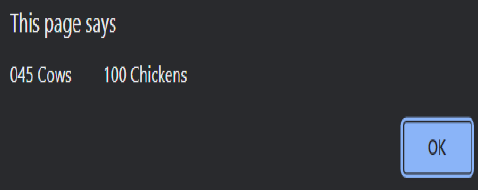
**Expected output:**

**007Cows 011Chickens**

**CODE**



**OUTPUT**

**  **

**JavaScript Objects and Regular Expressions**

**1. Write a JavaScript program to list the properties of a JavaScript**

**object.**

**Sample object:**

**var student = {**

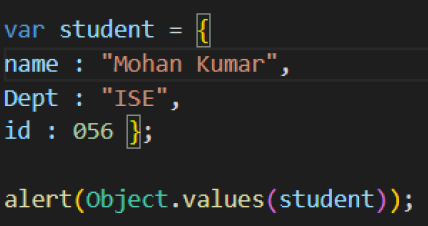
**name : "Mohan Kumar",**

**Dept : "ISE",**

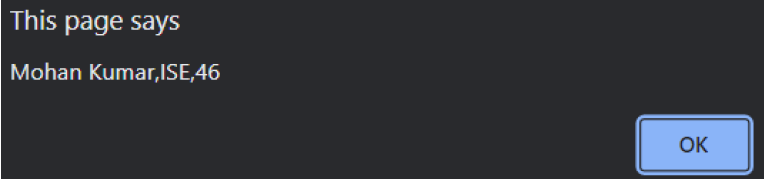
**id : 056 };**

**Sample Output: Mohan Kumar, ISE, 056**

**CODE**



**OUTPUT**

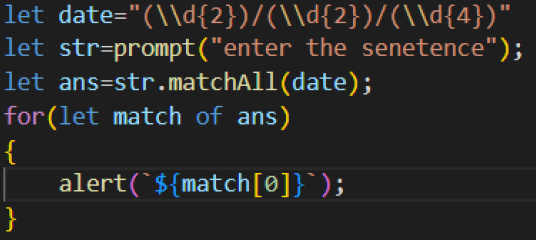


**2. Write a JavaScript program to search a date within a string.**

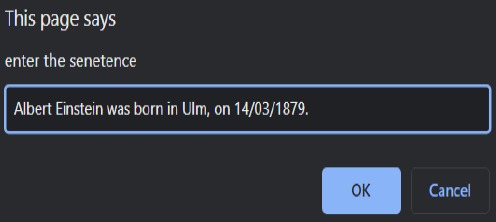
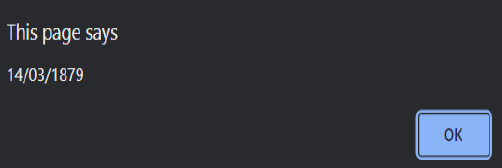
**Sample Input: "Albert Einstein was born in Ulm, on 14/03/1879."**

**Sample Output:14/03/1879.**

**CODE**

****

**OUTPUT**

 ****

**3. Write a pattern that matches e-mail addresses. Syntax:**

**localpart@domain**

**Note: The local part (The text before @ symbol) contains the following**

**ASCII characters.**

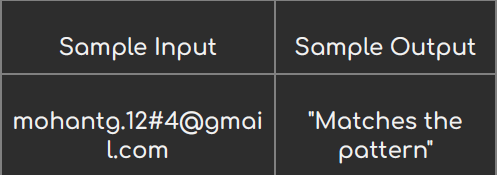
**Uppercase (A-Z) and lowercase (a-z) English letters.**

**Digits (0-9).**

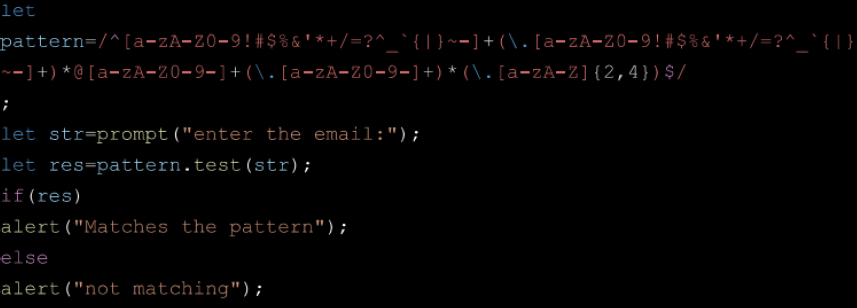
**Characters ! # $ % & ' \* + - / = ? ^ \_ ` { | } ~**

**Character . (dot) provided that it is not the first or last character**

**and it will not come one after the other.**

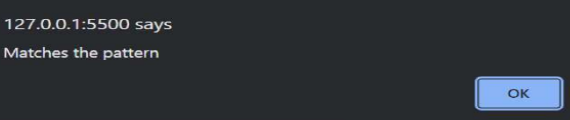
****

**CODE**

****

**OUTPUT**

****

****