19I510 Design and Analysis of Algorithms

Exercise 1

Analysis of Algorithms

Learning Objective

1. Devise algorithm for problem statements and implement in programming language of their choice
2. Use asymptotic analysis framework to assess the time complexity for algorithms

Exercise Problem Statements

1. Given a number n, print all primes smaller than or equal to n. It is also given that n is a small number. Adopt the Sieve of Eratosthenes approach for finding the numbers.
2. It uses a **temp dict** for storing all numbers from 2 to n along with a value **True**. It denotes initially all numbers are considered as prime
3. Traverse the dict from beginning to determine a number with value as True.

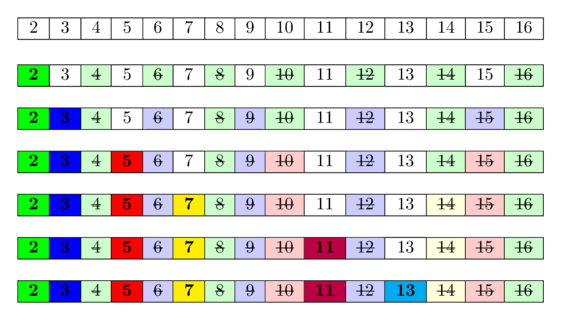
Let us consider the number as x.

X is considered as the prime number (Displayed)

Change the value of all numbers > n which are multiples of the X as False

1. Repeat the same until all the numbers are tested.

Visualization of the algorithm for computing numbers from 1 to 16 is shown in the figure below



**Input Format**The first line of input consists of value of n

**Output Format**The output will print all the primes in increasing order separated by a tab space

**Sample Input**

16

**Sample Output**  
2 3 5 7 11 13

1. Implement the Euclid’s algorithm for determining the greatest common divisor of two numbers
2. Let the numbers be a and b (Both a and b are greater than 0)

a>=b

1. Repeatedly perform the following operations until b reaches a value of 0

Replace a by b

Replace b by a mod b

**Input Format**The first line contains the number of test cases(n)

The following n lines contain pair of numbers separated by a space

**Output Format**Print the gcd of numbers for every pair in a separate line

**Sample Input**

2

25 17

3 12

**Sample Output**

1

3

1. Given an m by n grid of letters and a list of words, find the location in the grid at

which the word can be found.

A word can match the letters in the grid regardless of case (i.e., upper- and lowercase letters are to be treated as the same). The matching can be done in any of the eight (horizontal, vertical, or diagonal directions) through the grid. The match process is case-insensitive.

**Input Format**

The input begins with a single positive integer on a line by itself indicating the number of cases. Each case begins with a pair of integers m followed by n on a single line separated by a space. The next m lines contain n characters each. Following the grid of letters, another integer k is provided. The next k lines of input contain the list of words to search for, one word per line.

Constraints: 1 ≤ m, n ≤ 50

1 ≤k ≤ 20

**Output Format**For each word output a pair of integers representing its location in the corresponding grid. The integers must be separated by a single space. The first integer is the line in which the word is found and the second integer is the column in which the word is found.

If a word is found more than once in the grid, then output the location of the uppermost occurrence of the word. Closer to the top of the grid. If a word is not found print -1

**Sample Input**

1

8 11

abcDEFGhigg

hEbkWalDork

FtyAwaldORm

FtsimrLqsrc

byoArBeDeyv

Klcbqwikomk

strEBGadhrb

yUiqlxcnBjf

3

Waldorf

Bambi

Betty

**Sample Output**

2 5

2 3

1 2

1. Rio de Janeiro is a very beautiful city. But there are so many places to visit that sometimes you feel a bit lost. But your friend has promised to be a tourist guide provided that you help him find all the places where the police will fine the vehicle. As he is not a good driver himself, he wants to avoid fine at any cost.

Those cameras are strategically distributed over the city, in locations that a driver must pass through in order to go from one zone of the city to another. In order words, if there are two city locations A and B such that to go from one to another (A to B or B to A) a driver must pass through a location C, then C will have a camera. (These are the places where the vehicle can be fined)

Your task is to help Bruno to find all the cameras, given the map of the city, so he can be your tourist guide and avoid further fines.

**Input Format**

The first line contains the number of vertices (n). Following n lines contain the name of the cities. The next line contains the number of edges (E). Following E number of edges contain two cities separated by a space indicating a direct path connecting the cities

**Output Format**Print the number of cameras found along with the locations where the camera is found. If there is more than one location, print them in a lexicographically ordered format

**Sample Input**

6

St1

St2

St3

St4

St5

St6

7

St4 St3

St3 St1

St4 St1

St2 St6

St1 St2

St5 St1

St6 St5

**Sample Output**  
1

St1