**Consecutive Prime Sum**

Some prime numbers can be expressed as a sum of other consecutive prime numbers. For example 5 = 2 + 3, 17 = 2 + 3 + 5 + 7, 41 = 2 + 3 + 5 + 7 + 11 + 13. Your task is to find out how many prime numbers which satisfy this property are present in the range 3 to N subject to a constraint that summation should always start with number 2.

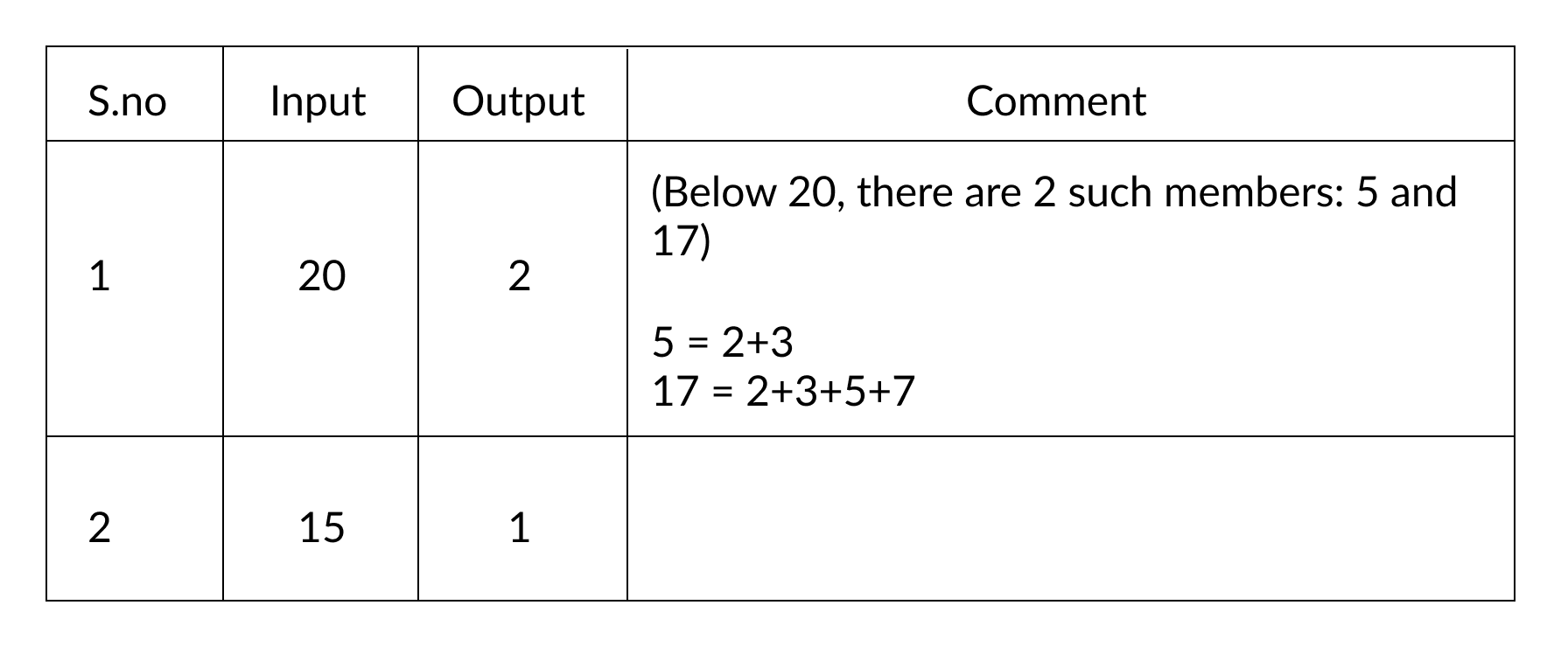
Write code to find out the number of prime numbers that satisfy the above-mentioned property in a given range.

**Input Format:** First line contains a number N

**Output Format:** Print the total number of all such prime numbers which are less than or equal to N.

**Constraints:** 2<N<=12,000,000,000

**Sample:**



**Possible solution:**

#include <stdio.h>  
int prime(int b)  
{  
    int j,cnt;  
   cnt=1;  
     for(j=2;j<=b/2;j++)  
     {  
         if(b%j==0)  
         cnt=0;  
     }  
     if(cnt==0)  
     return 1;  
     else  
     return 0;  
}  
int main() {  
 int i,j,n,cnt,a[25],c,sum=0,count=0,k=0;  
 scanf("%d",&n);  
 for(i=2;i<=n;i++)  
 {  
     cnt=1;  
     for(j=2;j<=n/2;j++)  
     {  
         if(i%j==0)  
         cnt=0;  
     }  
     if(cnt==1)  
     {  
        a[k]=i;  
        k++;  
        }  
 }  
 for(i=0;i<k;i++)  
 {  
     sum=sum+a[i];  
    c= prime(sum);  
    if(c==1)  
    count++;  
 }  
 printf("%d",count);  
 return 0;  
}

**Output:**

20  
2

## ****Bank Compare****

There are two banks – Bank A and Bank B. Their interest rates vary. You have received offers from both banks in terms of the annual rate of interest, tenure, and variations of the rate of interest over the entire tenure.

You have to choose the offer which costs you least interest and reject the other. Do the computation and make a wise choice.

The loan repayment happens at a monthly frequency and Equated Monthly Installment (EMI) is calculated using the formula given below :

EMI = loanAmount \* monthlyInterestRate / ( 1 – 1 / (1 + monthlyInterestRate)^(numberOfYears \* 12))

**Constraints:**

1 <= P <= 1000000  
1 <=T <= 50  
1<= N1 <= 30  
1<= N2 <= 30

**Input Format:**

* First line: P – principal (Loan Amount)
* Second line: T – Total Tenure (in years).
* Third Line: N1 is the number of slabs of interest rates for a given period by Bank A. First slab starts from the first year and the second slab starts from the end of the first slab and so on.
* Next N1 line will contain the interest rate and their period.
* After N1 lines we will receive N2 viz. the number of slabs offered by the second bank.
* Next N2 lines are the number of slabs of interest rates for a given period by Bank B. The first slab starts from the first year and the second slab starts from the end of the first slab and so on.
* The period and rate will be delimited by single white space.

**Output Format:**Your decision – either Bank A or Bank B.

**Explanation:**

Example 1

Input

10000  
20  
3  
5 9.5  
10 9.6  
5 8.5  
3  
10 6.9  
5 8.5  
5 7.9

Output: Bank B

Example 2

Input

500000  
26  
3  
13  9.5  
3  6.9  
10  5.6  
3  
14  8.5  
6  7.4  
6  9.6

Output: Bank A

**Possible solution:**

#include   
#includeint main() {  
 double p,s,mi,sum,emi,bank[5],sq;  
 int y,n,k,i,yrs,l=0;  
    scanf("%lf",&p);  
 scanf("%d",&y);  
 for(k=0;k<2;k++)  
 {  
 scanf("%d",&n);  
 sum=0;  
 for(i=0;i<n;i++)  
 {  
     scanf("%d",&yrs);  
     scanf("%lf",&s);  
     mi=0;  
     sq=pow((1+s),yrs\*12);  
     emi= (p\*(s))/(1-1/sq);  
     sum= sum + emi;  
 }

 bank[l++]=sum;  
 }  
 if(bank[0]<bank[1])  
 printf(“Bank A”);

 else  
 printf(“Bank B”);  
 return 0;  
}

## ****Counting Rock Samples****

Juan Marquinho is a geologist and he needs to count rock samples in order to send it to a chemical laboratory. He has a problem: The laboratory only accepts rock samples by a range of its size in ppm (parts per million).

Juan Marquinho receives the rock samples one by one and he classifies the rock samples according to the range of the laboratory. This process is very hard because the number of rock samples may be in millions.

Juan Marquinho needs your help, your task is to develop a program to get the number of rocks in each of the ranges accepted by the laboratory.

**Input Format:**

An positive integer S (the number of rock samples) separated by a blank space, and a positive integer R (the number of ranges of the laboratory); A list of the sizes of S samples (in ppm), as positive integers separated by space R lines where the ith line containing two positive integers, space separated, indicating the minimum size and maximum size respectively of the ith range.

**Output Format:**

R lines where the ith line containing a single non-negative integer indicating the number of the samples which lie in the ith range.

**Constraints:**10 ≤ S ≤ 10000 1 ≤ R ≤ 1000000 1≤size of each sample (in ppm) ≤ 1000

Example 1

Input: 10 2

           345 604 321 433 704 470 808 718 517 811

           300 350

           400 700

Output: 2 4

**Explanation:**

There are 10 samples (S) and 2 ranges ( R ). The samples are 345, 604,…811. The ranges are 300-350 and 400-700. There are 2 samples in the first range (345 and 321) and 4 samples in the second range (604, 433, 470, 517). Hence the two lines of the output are 2 and 4

Example 2

Input: 20 3

           921 107 270 631 926 543 589 520 595 93 873 424 759 537 458 614 725 842 575 195

           1 100

           50 600

           1 1000

Output: 1 12 20

**Explanation:**

There are 20 samples and 3 ranges. The samples are 921, 107 … 195. The ranges are 1-100, 50-600 and 1-1000. Note that the ranges are overlapping. The number of samples in each of the three ranges are 1, 12 and 20 respectively. Hence the three lines of the output are 1, 12 and 20.

**Possible Solution:**

#include   
int main() {  
 int a[1000],s,i,j,t,l1,l2,c=0;  
 scanf("%d",&s);  
 scanf("%d",&t);  
 for(i=0;i<s;i++)  
 scanf("%d",&a[i]);  
 for(i=0;i<t;i++)  
 {  
 scanf("%d %d",&l1,&l2);  
 for(j=0;j<s;j++)  
 {  
     if((a[j]>=l1)&&(a[j]<=l2))  
     c++;  
 }  
 printf("%d\n  ",c);  
 c=0;  
 }  
 return 0;  
}

**kth largest factor of N**

A positive integer d is said to be a factor of another positive integer N if when N is divided by d, the remainder obtained is zero. For example, for number 12, there are 6 factors 1, 2, 3, 4, 6, 12. Every positive integer k has at least two factors, 1 and the number k itself. Given two positive integers N and k, write a program to print the kth largest factor of N.

**Input Format:**The input is a comma-separated list of positive integer pairs (N, k)

**Output Format:** The kth highest factor of N. If N does not have k factors, the output should be 1.

**Constraints:** 1<N<10000000000. 1<k<600. You can assume that N will have no prime factors which are larger than 13.

Example 1

* Input: 12,3
* Output: 4

**Explanation:**N is 12, k is 3. The factors of 12 are (1,2,3,4,6,12). The highest factor is 12 and the third largest factor is 4. The output must be 4

Example 2

* Input: 30,9
* Output: 1

**Explanation:**N is 30, k is 9. The factors of 30 are (1,2,3,5,6,10,15,30). There are only 8 factors. As k is more than the number of factors, the output is 1.

**Possible Solution:**

#include <stdio.h>  
int main() {  
 int n,k,i,c=0;  
 scanf("%d",&n);  
 scanf("%d",&k);  
 for(i=n;i>=1;i--)  
 {  
     if((n%i)==0)  
     c++;  
     if(c==k)  
     {  
     printf("%d",i);  
     break;  
     }  
 }  
 if(c!=k)  
 printf("1");  
 return 0;  
}

**Collecting Candies**

Krishna loves candies a lot, so whenever he gets them, he stores them so that he can eat them later whenever he wants to.

He has recently received N boxes of candies each containing Ci candies where Ci represents the total number of candies in the ith box. Krishna wants to store them in a single box. The only constraint is that he can choose any two boxes and store their joint contents in an empty box only. Assume that there are infinite number of empty boxes available.

At a time he can pick up any two boxes for transferring and if both the boxes say contain X and Y number of candies respectively, then it takes him exactly X+Y seconds of time. As he is to eager to collect all of them he has approached you to tell him the minimum time in which all the candies can be collected.

**Input Format:**

* The first line of input is the number of test case T
* Each test case is comprised of two inputs
* The first input of a test case is the number of boxes N
* The second input is N integers delimited by whitespace denoting the number of candies in each box

**Output Format:**Print minimum time required, in seconds, for each of the test cases. Print each output on a new line.

**Constraints:**

* 1 ≤T≤10
* 1 ≤N≤ 10000
* 1 ≤ [Candies in each box] ≤ 100009

**Sample Input and Output:**

|  |  |  |
| --- | --- | --- |
| **SNo** | **Input** | **Output** |
| 1 | 1 4 1 2 3 4 | 19 |
| 2 | 1 5 1 2 3 4 5 | 34 |

**Explanation for sample input-output 1:**

4 boxes, each containing 1, 2, 3 and 4 candies respectively. Adding 1 + 2 in a new box takes 3 seconds. Adding 3 + 3 in a new box takes 6 seconds. Adding 4 + 6 in a new box takes 10 seconds. Hence total time taken is 19 seconds. There could be other combinations also, but overall time does not go below 19 seconds.

**Explanation for sample input-output 2:**

5 boxes, each containing 1, 2, 3, 4 and 5 candies respectively. Adding 1 + 2 in a new box takes 3 seconds. Adding 3 + 3 in a new box takes 6 seconds. Adding 4 + 6 in a new box takes 10 seconds. Adding 5 + 10 in a new box takes 15 seconds. Hence total time taken is 34 seconds. There could be other combinations also, but overall time does not go below 33 seconds.

**Possible Solution:**

#include <stdio.h>  
int main() {  
 int n,i,k=0,sum=0,s1=0,t,temp=0,j;  
 long c[100009],s[100009];  
 scanf("%d",&t);  
 for(int l=0;l<t;l++)  
 {  
 scanf("%d",&n);  
 for(i=0;i<n;i++)  
 scanf("%ld",&c[i]);  
 for(i=0;i<n;i++)  
 {  
 for(j=i+1;j<n;j++)  
 {  
 if(c[i]>c[j])  
 {  
 temp=c[i];  
 c[i]=c[j];  
 c[j]=temp;  
 }  
 }  
 }  
 sum=0;  
 k=0;  
 for(i=0;i<n;i++)  
 {  
 sum=sum+c[i];  
 s[k]=sum;  
 k++;  
 }  
 s1=0;  
 for(i=1;i<k;i++)  
 s1=s1+s[i];  
 printf("%d\n",s1);  
 }  
 return 0;  
}

**Football League**

Football League Table Statement : All major football leagues have big league tables. Whenever a new match is played, the league table is updated to show the current rankings (based on Scores, Goals For (GF), Goals Against (GA)). Given the results of a few matches among teams, write a program to print all the names of the teams in ascending order (Leader at the top and Laggard at the bottom) based on their rankings.

**Rules::** A win results in 2 points, a draw results in 1 point and a loss is worth 0 points. The team with the most goals in a match wins the match. Goal Difference (GD) is calculated as Goals For (GF) — Goals Against (GA). Teams can play a maximum of two matches against each other — Home and Away matches respectively.

The ranking is decided as follows: Team with maximum points is ranked 1 and minimum points is placed last Ties are broken as follows Teams with same points are ranked according to Goal Difference(GD).

If Goal Difference(GD) is the same, the team with higher Goals For is ranked ahead

If GF is same, the teams should be at the same rank but they should be printed in case-insensitive alphabetic according to the team names. More than 2 matches of same teams, should be considered as Invalid Input.

A team can’t play matches against itself, hence if team names are same for a given match, it should be considered Invalid Input

**Input Format:** First line of input will contain number of teams (N) Second line contains names of the teams (Na) delimited by a whitespace character Third line contains number of matches (M) for which results are available Next M lines contain a match information tuple {T1 T2 S1 S2}, where tuple is comprised of the following information

* T1 — Name of the first team
* T2 — Name of the second team
* S1 — Goals scored by the first team
* S2 — Goals scored by the second team

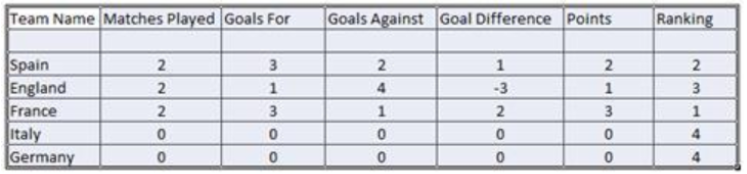
**Output Format:** Team names in order of their rankings, one team per line OR Print “Invalid Input” where appropriate.

**Constraints:** 0< N <=10,000 0<=S1,S2

**Example:** Consider 5 teams Spain, England, France, Italy and Germany with the following fixtures:

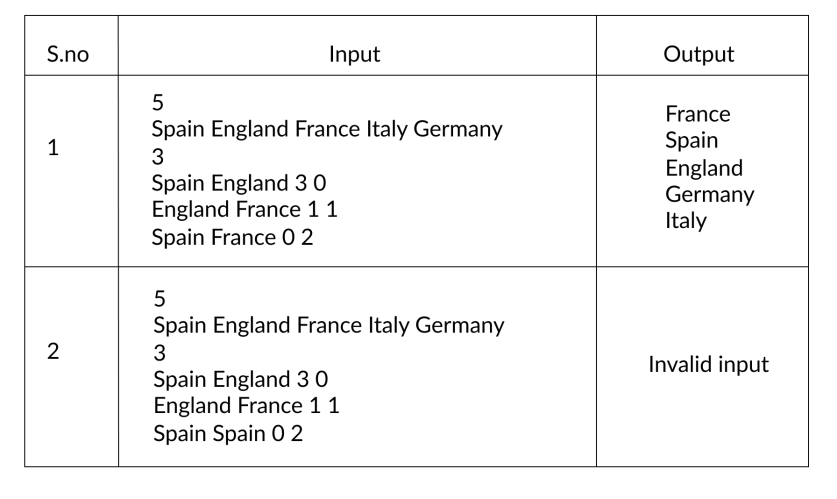
* Match 1: Spain vs. England (3-0) (Spain gets 2 points, England gets 0)
* Match 2: England vs. France (1-1) (England gets 1 point, France gets 1)
* Match 3: Spain vs. France (0-2) (Spain gets 0 points, France gets 2)

Table 1. Points Table after 3 matches



Since, Italy and Germany are tied for points, goals difference is checked. Both have same, so, Goals For is checked. Since both are same. Germany and Italy share the 4th rank. Since Germany appears alphabetically before Italy, Germany should be printed before Italy. Then the final result is: France Spain England Germany Italy

Sample:



## ****Sorting Boxes****

The parcel section of the Head Post Office is in a mess. The parcels that need to be loaded to the vans have been lined up in a row in an arbitrary order of weights. The Head Post Master wants them to be sorted in the increasing order of the weights of the parcels, with one exception.  He wants the heaviest (and presumably the most valuable) parcel kept nearest his office.

You and your friend try to sort these boxes and you decide to sort them by interchanging two boxes at a time. Such an interchange needs effort equal to the product of the weights of the two boxes.

The objective is to reposition the boxes as required with minimum effort.

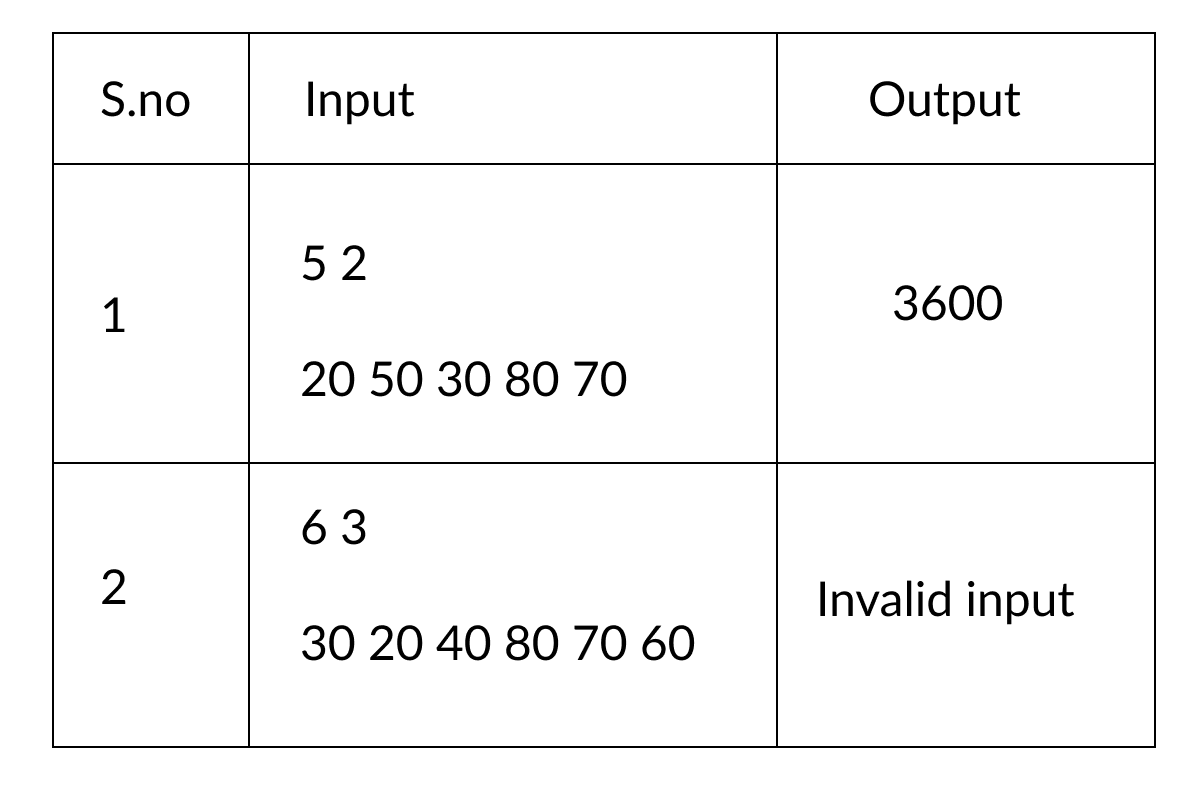
**Input Format:** The first line consists of two space-separated positive integers giving the number of boxes (N) and the position of the Head Post Masters office (k) where the heaviest box must be.

The second line consists of N space separated positive integers giving the weights of the boxes. You may assume that no two weights are equal.

**Output Format:** The output is one line giving the total effort taken to get the boxes in sorted order, and the heaviest in position k.

**Constraints:**N<=50 and Weights <= 1000

**Sample Input-output:**



**On A Cube**

A solid cube of 10 cm x 10cm x 10 cm rests on the ground. It has a beetle on it, and some sweet honey spots at various locations on the surface of the cube. The beetle starts at a point on the surface of the cube and goes to the honey spots in order along the surface of the cube.

* If it goes from a point to another point on the same face (say X to Y), it goes in an arc of a circle that subtends an angle of 60 degrees at the center of the circle
* If it goes from one point to another on a different face, it goes by the shortest path on the surface of the cube, except that it never travels along the bottom of the cube

The beetle is a student of cartesian geometry and knows the coordinates (x, y, z) of all the points it needs to go to. The origin of coordinates it uses is one corner of the cube on the ground, and the z-axis points up.  Hence, the bottom surface (on which it does not crawl) is z=0, and the top surface is z=10.  The beetle keeps track of all the distances traveled, and rounds the distance traveled to two decimal places once it reaches the next spot so that the final distance is a sum of the rounded distances from spot to spot.

**Input Format:**The first line gives an integer N, the total number of points (including the starting point) the beetle visits

The second line is a set of 3N comma separated non-negative numbers, with up to two decimal places each. These are to be interpreted in groups of three as the x, y, z coordinates of the points the beetle needs to visit in the given order.

**Output Format:**One line with a number giving the total distance traveled by the beetle accurate to two decimal places. Even if the distance traveled is an integer, the output should have two decimal places.

**Constraints:**None of the points the beetle visits is on the bottom face (z=0) or on any of the edges of the cube (the lines where two faces meet)

2<=N<=10

**Sample Input-Output:**

Input  
3  
1,1,10,2,1,10,0,5,9  
Output  
6.05

**Input**  
3  
1,1,10,2,1,10,0,1,9  
**Output**  
4.05

## Bride Hunting

**Problem Description**

Sam is an eligible bachelor. He decides to settle down in life and start a family. He goes bride hunting.

He wants to marry a girl who has at least one of the 8 qualities mentioned below:-

1) The girl should be rich.

2) The girl should be an Engineer/Doctor.

3) The girl should be beautiful.

4) The girl should be of height 5.3".

5) The girl should be working in an MNC.

6) The girl should be an extrovert.

7) The girl should not have spectacles.

8) The girl should be kind and honest.

He is in search of a bride who has some or all of the 8 qualities mentioned above. On bride hunting, he may find more than one contenders to be his wife.

In that case, he wants to choose a girl whose house is closest to his house. Find a bride for Sam who has maximum qualities. If in case, there are more than one contenders who are at equal distance from Sam’'s house; then

print "“Polygamy not allowed”".

In case there is no suitable girl who fits the criteria then print “**"No suitable girl found"**”

Given a Matrix N\*M, Sam's house is at (1, 1). It is denoted by 1. In the same matrix, the location of a marriageable Girl is also denoted by 1. Hence 1 at location (1, 1) should not be considered as the location of a marriageable Girl’s location.

The qualities of that girl, as per Sam’'s criteria, have to be decoded from the number of non-zero neighbors (max 8-way) she has. Similar to the condition above, 1 at location (1, 1) should not be considered as the quality of a Girl. See Example section to get a better understanding.

Find Sam, a suitable Bride and print the row and column of the bride, and find out the number of qualities that the Bride possesses.

NOTE: - Distance is calculated in number of hops in any direction i.e. (Left, Right, Up, Down and Diagonal)

**Constraints**

2 <= N,M <= 10^2

**Input Format**

First Line contains the row (N) and column (M) of the houses.

Next N lines contain the data about girls and their qualities.

**Output**

It will contain the row and column of the bride, and the number of qualities that Bride possess separated by a colon (i.e. :).

**Explanation**

Example 1

**Input**:

2 9

1 0 1 1 0 1 1 1 1

0 0 0 1 0 1 0 0 1

**Output**:

1:7:3

**Explanation**:

The girl and qualities are present at (1,3),(1,4),(1,6),(1,7),(1,8),(1,9),(2,4),(2,6),(2,9).

The girl present at (1,3) has 2 qualities (i.e. (1,4)and (2,4)).

The girl present at (1,4) has 2 qualities.

The Bride present at (1,6) has 2 qualities.

The Bride present at (1,7) has 3 qualities.

The Bride present at (1,8) has 3 qualities.

The Bride present at (1,9) has 2 qualities.

The Bride present at (2,4) has 2 qualities.

The Bride present at (2,6) has 2 qualities.

The Bride present at (2,9) has 2 qualities.

As we see, there are two contenders who have maximum qualities, one is at (1,7) and another at (1,8).

The girl who is closest to Sam's house is at (1,7). Hence, she is the bride.

Hence, the output will be 1:7:3.

Example 2

**Input**:

6 6

1 0 0 0 0 0

0 0 0 0 0 0

0 0 1 1 1 0

0 0 1 1 1 0

0 0 1 1 1 0

0 0 0 0 0 0

**Output**:

4:4:8

**Explanation**:

The bride and qualities are present at (3,3),(3,4),(3,5),(4,3),(4,4),(4,5),(5,3),(5,4),(5,5)

The Bride present at (3,3) has 3 qualities (i.e. (3,4),(4,3) and (4,4)).

The Bride present at (3,4) has 5 qualities.

The Bride present at (3,5) has 3 qualities.

The Bride present at (4,3) has 5 qualities.

The Bride present at (4,4) has 8 qualities.

The Bride present at (4,5) has 5 qualities.

The Bride present at (5,3) has 3 qualities.

The Bride present at (5,4) has 5 qualities.

The Bride present at (5,5) has 3 qualities.

As we see, the girl present in (4,4) has maximum number of Qualities. Hence, she is the bride.

Hence, the output will be 4:4:8.

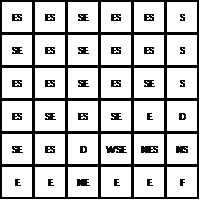
## Skateboard

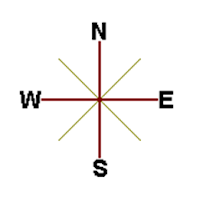
**Problem Description**

The amusement park at Patagonia has introduced a new skateboard competition. The skating surface is a grid of N x N squares. Most squares are so constructed with slopes that it is possible to direct the skateboard in any of up to three directions of the possible four (North ,East, South or West, represented by the letters N, E, S and W respectively). Some squares however have a deep drop from the adjacent square from which it is impossible to go to any adjacent square. These are represented by D (for Drop) in that square. The objective is to maneuver the skateboard to reach the South East corner of the grid, marked F.

Each contestant is given a map of the grid, which shows where the Drop squares are (marked D), where the Final destination is (marked F), and, for each other square, the directions it is possible to maneuver the skateboard in that square.

The contestant draws lots to determine which of the squares on the boundaries of the grid on the North or the West of the grid (the top or the left in the diagram) he or she should start in. Then, using a map of the grid, he or she needs to try to reach the South East corner destination by maneuvering the skateboard.

[](https://1.bp.blogspot.com/-w3FiEsZWqaw/W2RtZy4IyZI/AAAAAAAAfuU/sIgitHYqd1QHPiseCHLOG4r3ZG60UXn3QCLcBGAs/s1600/Skateboardimage1.png)

. [](https://3.bp.blogspot.com/-bjcxriNfsRE/W2RtigrQoaI/AAAAAAAAfuY/01JeZWipaAIIspikRp5GCta80R_obMWOwCLcBGAs/s1600/Skateboardimage2.png)

In some cases, it is impossible to reach the destination. For example, in the diagram above, if one starts at the North East corner (top right in the diagram), the only way is to go is South, until the Drop square is reached (three squares South), and the contestant is stuck there.

A contestant asks you to figure out the number of squares at the North or West boundary (top or left boundary in the map) from which it is feasible to reach the destination.

**Constraints**

5<=N<=50

**Input Format**

The first line of the input is a positive integer N, which is the number of squares in each side of the grid.

The next N lines have a N strings of characters representing the contents of the map for that corresponding row. Each string may be F, representing the Final destination, D, representing a drop square, or a set of up to three of the possible four directions (N,E,S,W) in some random order. These represent the directions in which the contestant can maneuver the skateboard when in that square.

**Output**

The output is one line with the number of North or West border squares from which there is a safe way to maneuver the skateboard to the final destination.

**Explanation**

**Example 1**

Input

6

ES,ES,SE,ES,ES,S

SE,ES,SE,ES,ES,S

ES,ES,SE,ES,SE,S

ES,SE,ES,SE,E,D

SE,ES,D,WSE,NES,NS

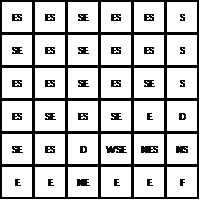
E,E,NE,E,E,F

Output

9

Explanation

N =6, and the size of the grid is 6x6. The map of the grid is as below.

[](https://3.bp.blogspot.com/-DvOSZd4STT8/W2Rt-Us8czI/AAAAAAAAfuk/MAVdelH7kesTHYgSmxEqYbGRr3k4fSSWgCLcBGAs/s1600/Skateboardimage1%2B%25281%2529.png)

There are many ways to maneuver the skateboard. For example, if the contestant starts from the North West corner (top left in the map) and goes East all the way until he reaches the top right corner in the map, and then goes South, he will reach a Drop square. But if he goes South all the way from the same square until he reaches the bottom left square on the map, and keeps going East from there, the Final destination will be reached. Hence the top left square (1,1) is safe.

Similarly, from the square (1,5), all the paths lead to a drop square., The other 9 North and West border squares have ways skateboard to get to the final destination. The output is 9

**Example 2**

Input

5

ES,SE,ES,SE,S

S,EWS,SE,E,S

E,D,SEW,NSE,S

NE,N,SEW,D,W

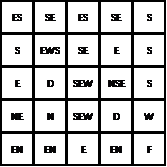
EN,EN,E,EN,F

Output

4

Explanation

N=5, and the grid is 5 x 5 squares. The map of the grid looks like this.

[](https://4.bp.blogspot.com/-zex0KEm6LnQ/W2RuJTw0i4I/AAAAAAAAfuo/W0-9qmZmO-APlP7S4L42-jr0NPl6mVUsgCLcBGAs/s1600/Skateboardimage3.png)

It can be seen that from squares (1,4) and (1,5), there is no way to maneuver the skateboard to the Final destination, and hence are not safe starting points.. Similarly, squares (2,1),(3,1), and (4,1) are not safe starting points. The only safe starting points on the North and West borders are (1,1),(1,2),(1,3),(5,1). Hence the output is 4

ESESSEESESS

SEESSEESESS

ESESSEESSES

ESSEESSEED

SEESDWSENESNS

EENEEEF

ESSEESSES

SEWSSEES

EDSEWNSES

NENSEWDW

ENENEENF

**Problem Description**

Rotate a given String in the specified direction by specified magnitude.

After each rotation make a note of the first character of the rotated String, After all rotation are performed the accumulated first character as noted previously will form another string, say **FIRSTCHARSTRING**.

Check If **FIRSTCHARSTRING** is an Anagram of any substring of the Original string.

If yes print "YES" otherwise "NO"**.**Input

The first line contains the original string s. The second line contains a single integer q. The ith of the next q lines contains character d[i] denoting direction and integer r[i] denoting the magnitude.

**Constraints**

1 <= Length of original string <= 30

1<= q <= 10

**Output**

YES or NO

**Explanation**

Example 1

Input

carrace

3

L 2

R 2

L 3

Output

NO

Explanation

After applying all the rotations the **FIRSTCHARSTRING** string will be "rcr" which is not anagram of any sub string of original string "carrace".