TCS CODEVITA PAST 3

PROBLEM1

Some prime numbers can be expressed as Sum of other consecutive prime numbers.
For example

5 = 2 + 3
17 = 2 + 3 + 5 + 7
23 = 7 + 11 + 13
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.

Write code to find out 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.

Sample Input1:

20

Sample Output1:

2

Sample Input2:

15

Sample Output2:

1

SNo	Name	Input	Output
1	TC8	100	11
2	TC6	40	4
3	TC2	15	1
4	TC4	25	3
5	TC10	90	10
6	TC7	49	5
7	TC5	36	4
8	TC3	5	1
9	TC9	99	11
10	TC1	20	2

```
#include<stdio.h>
int main()
{
int n,a[10000],i,j,c=0,k=0,c1=0,sum=0,temp,p,z,c2=0,s[100],h=0,temp1;
scanf("%d",&n);
for(i=2;i<=n;i++)
{
   c=0;
   for(j=1;j<=i;j++)
   {
       if(i%j==0)
       c++;
   }
```

```
   if(c==2)
   {
       a[k++]=i;
   }
}
k--;
c2=2;

for(i=0;i<k;i++)
{
   for(j=0;j<k;j++)
        {
            temp=1;
            for(z=j;temp<=c2;z++)
            {

               sum=sum+a[z];
                temp++;
            }
                for(p=0;p<n;p++)
       {
           //printf("\n%d",a[p]);
           //printf("%d ",a[j]);
           if(a[p]==sum)
           {
               s[h]=a[p];
               h++;
           }
       }
       sum=0;
        }
        c2++;

}

for(i=0;i<h;i++)
{
    for(j=i+1;j<h;j++)
    {
        if(s[i]>s[j])
        {
```

```
            temp1=s[i];
            s[i]=s[j];
            s[j]=temp1;
        }
    }
}
for(i=0;i<h;i++)
{
    if(s[i]!=s[i+1])
        c1++;
}


printf("%d",c1);

return 0;
}
```

PROBLEM 2

There are several piles of cards on the table arranged from left to right. All of them do not contain the same number of cards. As the boss insists on symmetry, Ramu is asked to merge the minimum number of adjacent piles so that if a list of the number of cards in each pile is given in left to right order, the list reads the same whether it is read in the left to right order or the right to left order.

For example, if the piles of cards initially were 1 2 2 5 1 3 1 1, there can be three merges (2,2), (5,1) and (3,1) to give 1 4 6 4 1.
Note that merging 3 numbers, say (1,2,2) is counted as 2 merges, one between (1 2) and the next between the resulting 3 and the next 2.
Hence another solution (1,2,2) (5,1) (3,1,1) resulting in 5 6 5 takes 5 merges(2+1+2), and is not minimal.

Another (trivial) solution is to merge all the piles to give 16. This may be the only solution in some cases, and shows that all initial sets have at least one (not necessarily minimal) solution

Input

There are two lines.
The first line is the number of piles.
The next line contains space separated positive integers, representing the numbers of cards in the piles from left to right.

Output

If the minimal merge results in more than one pile, the output has two lines, the first giving one integer (the number of merges) and the second containing space separated integers, giving the number of cards in the final pile sequence in order (left to right).
If there is no solution other than merging all into one pile, the output will be a single line containing the letters "TRIVIAL MERGE".

Constraints

The initial number of piles will be less than 50, and the number of cards in each pile in the initial configuration will be less than 1000.
Example 1

Input:
8
1 4 3 6 1 2 1 5

Output:
3
5 3 7 3 5

Explanation:
If we merge (1 4) , (6 1) and (2 1), (or three merges) the resulting set of piles is 5 3 7 3 5 (which is the same when read from left to right or right to left). As this is the minimum set of merges, the output is 3 (the number of merges) in the first line and 5 3 7 3 5 (the final pile sequence) in the second line.
Example 2

Input:
10
1 7 6 2 5 9 7 4 2 8

Output:
3
8 6 7 9 7 6 8

Explanation:
If we merge (1 7) , (2 5) and (4 2), (or three merges) the resulting set of piles is 8 6 7 9 7 6 8 (which is the same when read from left to right

or right to left). As this is the minimum set of merges, the output is 3 (the number of merges) and 5 3 7 3 5 (the final pile sequence).

Example 3

Input:
10
17 3 15 10 13 18 3 7 20 2

Output:
TRIVIAL MERGE

Explanation:
As there is no set of adjacent merges that results in a set of piles that reads the same both ways other than merging all into one pile (9 merges resulting in a pile with 108 cards), the output is one line, with "TRIVIAL MERGE" in it.

SNo	Input	Output
1	5 1 2 3 4 5	TRIVIAL MERGE
2	6 8 4 2 2 7 1	2 8 4 4 8
3	10 17 3 15 10 13 18 3 7 20 2	TRIVIAL MERGE
4	6 10 11 12 13 11 10	1 10 11 25 11 10
5	3 4 2 4	0 4 2 4

```
#include<stdio.h>
```

```
int main()
{
        int n,i,j,a[100],b=0,c,s[100],x=0,m=0;
        scanf("%d",&n);
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
        i=0;j=n-1,c=j;
        while(i<=j)
        {
            if(i==j)
            {
              s[b++]=a[i];
              break;
            }
            if(a[i]==a[j])
            {
                s[b++]=a[i++];
                s[c--]=a[j--];
                m++;
            }
            else if(a[i]>a[j])
            {
                j--;
                a[j]=a[j]+a[j+1];
                x++;
            }
            else if(a[i]<a[j])
            {
                i++;
                a[i]=a[i]+a[i-1];
                x++;
            }
        }
        if(x<=n/2)
        {
        printf("%d\n",x);
        for(i=0;i<b;i++)
        printf("%d ",s[i]);
        for(i=n-m;i<n;i++)
        printf("%d ",s[i]);
        }
```

```
        else
        printf("TRIVIAL MERGE");
        return 0;
}
```

PROBLEM 3

Calculate the **Final Salary** & **Final Accumulated PF** of an Employee working in ABC Company Pvt. Ltd. The Company gives two Increments (i.e. ***Financial Year Increment & Anniversary Increment***) to an Employee in a Particular Year.

Anniversary Increment is the increment for the employee after completing one year from the date of joining.
Financial Year Increment is the increment for the employee in the month of April.

The Employee must have Completed 1 Year to be Eligible for the Financial Year Increment. The Employee who are joining in the month of Financial Year Change (i.e. April) are considered as the Luckiest Employee's, because after completion of 1 Year, they get Two Increments
(Financial Year Increment & Anniversary Increment).

Rate of Interest for the Financial Year Increment = 11%.
Rate of Interest for the Anniversary Increment = 12%.

From 4th Year, the Financial Year Increment will be revised to 9%.
From 8th Year, the Financial Year Increment will be revised to 6%.
The Company is giving special Increment for the Employee who have completed 4 years & 8 years respectively.

Since the salary is varying for every anniversary/financial year increment, the accumalated PF is calculated by finding PF for every month and taking average of it for a single year(12months).

So, the Anniversary Increment of the Employee for the 4th Year will be 20% and the Anniversary Increment of the Employee for the 8th year will be 15%.

Calculate the Final Salary after N number of Years as well as Calculate the Accumulated PF of the Employee after N number of Years.

Please Note that, the Rate of Interest for calculating PF for a Particular Month is 12%. Moreover, take the upper Limit of the amount if it is in decimal (For e.g. - If any Amount

turns out to be 1250.02, take 1251 for the Calculation.)

Input Format:

1. Joining Date in dd/mm/yy format
2. Current CTC.
3. Number of Years for PF & Salary Calculation.

Output Format:

1. Salary after the Specified Number of Years (i.e. CTC after N number of Years) in the following format
   Final Salary = final salary amount
2. Accumulated PF of the Employee after N number of Years in the following format
   Final Accumulated PF = final accumulated PF amount

Constraints:

1. **Calculation should be done upto 11 digit precision and output should be printed with ceil value**

Sample Input:

01/01/2016

10000

2

Sample Output:

Final Salary = 13924

Final Accumulated PF = 2655

SNo	Input	Output
1	19/01/2016 6500 4	Final Salary = 14718 Final Accumulated PF = 4343
2	01/01/2016 10000 2	Final Salary = 13924 Final Accumulated PF = 2655
3	21/06/1997 2500 6	Final Salary = 8437 Final Accumulated PF = 3126
4	01/04/2000 2000 1	Final Salary = 2460 Final Accumulated PF = 240
5	01/04/2000 1000 4	Final Salary = 2401 Final Accumulated PF = 673
6	01/03/2000 1900 7	Final Salary = 7827 Final Accumulated PF = 3360
7	01/01/2013 10000 9	Final Salary = 59617 Final Accumulated PF = 28908

PROBLEM 4

In the strong room of ABC bank there are N vaults in a row. The amount of money inside each vault displayed on the door. You can empty any number of vaults as long as you do not empty more than 2 out of any 5 adjacent vaults. For example, of the vaults 1, 2, 3, 4, 5, if you empty 4 and 5, then you can not empty any of the vaults 6, 7 or 8 but you can empty 9th vault. If you attempt to break more than 2 of any 5 adjacent vaults, an alarm sounds and the sentry, a sharp shooter will kill you instantly with his laser gun!

The output is the maximum amount of money that can be emptied without sounding the alarm

Input

The first line contains an integer N which is the number of vaults. The next line has a sequence of positive integers of length N, giving the amount of cash in the vaults in order

Output

The maximum amount of money that can be looted without sounding the alarm.

Constraints

N≤50, Amount in each vault ≤50000

Example 1

Input:
9
1000,2000,1000,5000,9000,5000,3000,4000,1000

Output:
15000

Explanation:
One possible set of vaults to be looted to get the maximum possible are vaults 4, 5, 9 are looted, giving a total loot of (5000+9000+1000)=15000. Hence the output is 15000.

Example 2

Input:
10
5000,7000,3000,5000,9000,7000,6000,4000,7000,5000

Output:
28000

Explanation:
One possible set of vaults to be looted are (2, 5, 9, 10) Note that no set of five consecutive vaults has more than two vaults looted. The total looted is (7000 + 9000 + 7000 + 5000=28000). The out put is hence 28000.

SNo	Input	Output
1	9 1000,2000,1000,5000,9000,5000,3000,4000,1000	15000
2	9 1100,7200,4150,2500,8100,9500,6200,3500,6300	24100
3	10 1000,2000,3000,4000,5000,6000,7000,8000,9000,10000	28000
4	10 5000,7000,3000,5000,9000,7000,6000,4000,7000,5000	28000
5	9 1000,9000,8000,1000,2000,7000,6000,2000,3000	26000
6	9 3100,10000,1500,5000,2300,4000,7000,1000,900	24100

SNo	Input	Output
7	5 30000,50000,10000,20000,40000	90000
8	10 20000,19000,18000,17000,16000,15000,14000,13000,12000,11000	68000

PROBLEM 5

Given an MxN matrix, with a few hurdles arbitrarily placed, calculate the cost of longest possible route from point A to point B within the matrix.

Input Format:

1. First line contains 2 numbers delimited by whitespace where, first number M is number of rows and second number N is number of columns
2. Second line contains number of hurdles H followed by H lines, each line will contain one hurdle point in the matrix.
3. Next line will contain point A, starting point in the matrix.
4. Next line will contain point B, stop point in the matrix.

Output Format:

Output should display the length of the longest route from point A to point B in the matrix.

Examples :

Input :

3 10
3
1 2
1 5
1 8
0 0
1 7

Output :

24

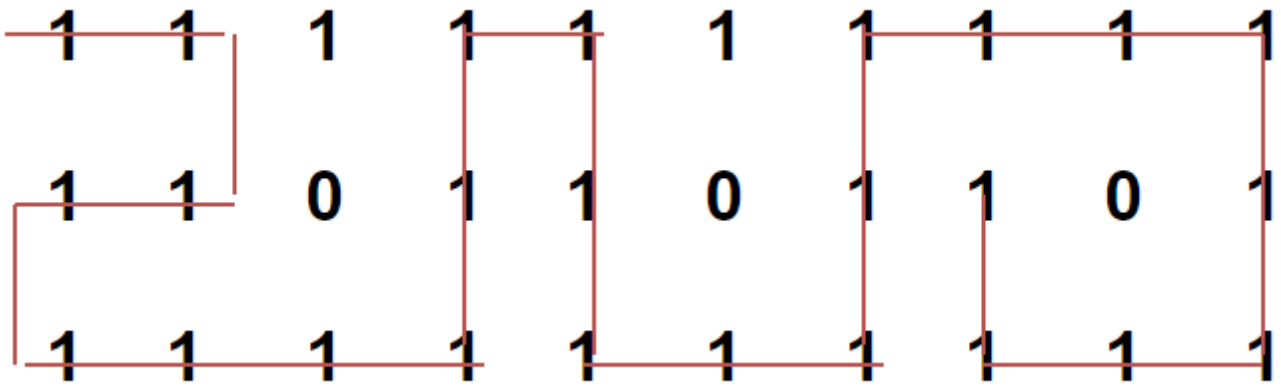
Here matrix will be of size 3x10 matrix with a hurdle at (1,2),(1,5) and (1,8) with starting point A(0,0) and stop point B(1,7)

3 10
3 -- (no. of hurdles)
1 2
1 5
1 8
0 0 -- (position of A)
1 7 -- (position of B)

So if you examine matrix below shown in **Fig 1**, total hops

(->) count is 24. So final answer will be 24. No other route longer than this one is possible in this matrix.

Fig 1:



Constraints:

1. The cost from one position to another will be 1 unit.

2. A location once visited in a particular path cannot be visited again.
3. A route will only consider adjacent hops. The route cannot consist of diagonal hops.
4. The position with a hurdle cannot be visited.
5. The values MxN signifies that the matrix consists of rows ranging from 0 to M-1 and columns ranging from 0 to N-1.
6. If the destination is not reachable or source/ destination overlap with hurdles, print cost as -1

Sample Input1:

```
3 10
3
1 2
1 5
1 8
0 0
1 7
```

Sample Output1:

```
24
```

Sample Input2:

```
2 2
1
0 0
1 1
0 0
```

Sample Output2:

```
-1
```

SNo	Input	Output
1	3 10 6 1 2 1 5 1 8 0 8 0 7 0 9 0 0 1 7	18
2	2 2 1 0 0 1 1 0 0	-1
3	5 10 6 1 2 1 5 1 8 0 8 0 7 0 9 0 0 1 7	42
4	5 5 4 1 3 2 2 2 4 4 4	16

SNo	Input	Output
	4 0 0 4	
5	5 5 9 1 0 1 4 1 1 1 2 1 4 1 3 2 2 2 4 4 4 4 0 0 4	-1
6	3 10 3 1 2 1 5 1 8 0 0 1 7	24
7	2 2 1 1 1 0 1 1 0	2

PROBLEM 6

A beetle is jumping on a checkerboard of size N X N squares. Each square has coordinates, a pair of integers (i,j), where i is the row number and j is the column number.

Associated with each square is the coordinates of the square it jumps to when it lands there. For example, in the 6 X 6 checkerboard below, the beetle jumps to (2,3) from (1,1), and jumps to (5,2) from (6,4)

If the starting location is given, the objective is to determine the position of the beetle after a large number of jumps.
Input
The first line of the input has three comma separated positive integers N,P,Q. N is the length of a side of the checkerboard (the checkerboard is of size N X N). The number of jumps the beetle makes is PQ (P multiplied by Q) The next N lines consist of N pairs of comma separated positive integers, each pair separated by a semicolon(;). The mth pair in line k represents the coordinates of the square it jumps to if it lands on square (k,m).
Finally, there is one line with a comma separated pair of numbers giving the initial position of the beetle
Output
The output is the coordinates of the beetle's position after PQ moves
Constraints
6<=N<=20 1<=P,Q<109

Example 1
Input:
6,2,3
2,3;2,4;2,1;3,5;3,4;4,2
4,2;4,1;3,1;3,6;4,4;1,4
1,2;1,3;4,5;5,5;2,1;1,5
6,2;6,1;2,2;5,6;2,6;2,5
3,2;3,3;6,5;6,6;6,3;6,4
5,3;5,4;5,1;5,2;4,6;1,6
1,2
Output:
6,3
Explanation: N is 6, P is 2 and Q is 3. The size of the checkerboard is 6 X 6. The next 6 lines give the "jump to" coordinates if the beetle is on the corresponding square. The starting position of the beetle is (1,2)
The checkerboard is the same as pictured above. The output should be the position of the beetle after (2)(3)=6 moves. The position after each of the moves is (2,4);(3,6);(1,5);(3,4);(5,5);(6,3). Hence the output is 6,3

Example 2

Input:
6,12,2
2,3;2,4;2,1;3,5;3,4;4,2
4,2;4,1;3,1;3,6;4,4;1,4
1,2;1,3;4,5;5,5;2,1;1,5
6,2;6,1;2,2;5,6;2,6;2,5
3,2;3,3;6,5;6,6;6,3;6,4
5 ,3;5,4;5,1;5,2;4,6;1,6
4 ,3

Output:
6,1

SNo	Input	Output
1	6,4,2 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 1,1	5,5
2	6,12,2 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 1,2	1,4

SNo	Input	Output
3	6,12,2 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 4,3	6,1
4	6,9,4 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 4,3	1,4
5	6,12,2 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 1,1	3,3
6	6,2,3 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 1,2	6,3

SNo	Input	Output
7	6,14,6 2,3;2,4;2,1;3,5;3,4;4,2 4,2;4,1;3,1;3,6;4,4;1,4 1,2;1,3;4,5;5,5;2,1;1,5 6,2;6,1;2,2;5,6;2,6;2,5 3,2;3,3;6,5;6,6;6,3;6,4 5,3;5,4;5,1;5,2;4,6;1,6 4,3	1,4

```
#include<stdio.h>
int main()
{
        int p,m,n,i,j,a,b,c[100][100],d[100][100],x,y,ct=0;
        scanf("%d,%d,%d",&p,&m,&n);
        for(i=1;i<=p;i++)
        {
            for(j=1;j<=p;j++)
            {
                scanf("%d,%d;",&c[i][j],&d[i][j]);
           //printf("%d %d",c[i][j],d[i][j]);
            }
            printf("\n");
        }
        scanf("%d,%d",&x,&y);
        while(ct<(m*n))
        {
            a=c[x][y];
            b=d[x][y];

            x=a;y=b;
            ct++;
        }
```

```
        printf("%d,%d",a,b);

        return 0;
}
```