Base 6

Problem Description

Given a sequence of distinct numbers a1, a2, ….. an, an inversion occurs if there are indices i<j such that ai > aj .

For example, in the sequence 2 1 4 3 there are 2 inversions (2 1) and (4 3).

The input will be a main sequence of N positive integers. From this sequence, a **Derived Sequence** will be obtained using the following rule. The output is the number of inversions in the derived sequence.

**Rule for forming derived sequence**

An integer may be represented in base 6 notation. In base 6, 10305 is 1x64 + 3x62 + 5 =1409. Note that none of the digits in that representation will be more than 5.

1296

108

5

The sum of digits in a base 6 representation is the sum of all the digits at the various positions in the representation. Thus for the number 1409, the representation is 10305, and the sum of digits is 1+0+3+0+5=9. The sum of digits may be done in the decimal system, and does not need to be in base 6

The derived sequence is the sum of the digits when the corresponding integer is represented in the base 6 form number will be expressed in base 6, and the derived sequence is the sum of the digits of the number in the base 6 representation.

Constraints

N <= 50

Integers in sequence <= 107

Input Format

The first line of the input will have a single integer, which will give N.

The next line will consist of a comma separated string of N integers, which is the main sequence

Output

The number of inversions in the derived sequence formed from the main sequence.

Explanation

**Example 1**

Input

5

55, 53, 88, 27, 33

Output

2

**Explanation**

The number of integers is 5, as specified in the first line. The given sequence is 55, 53, 88, 27, 33.

The base 6 representation is 131, 125, 224, 43, 53 The derived sequence is 5,8,8,7,8 (corresponding to the sum of digits). The number of inversions in this is 2, namely (8, 7), (8, 7)

**Example 2**

Input

|  |
| --- |
| 8 |
| 120,21,47,64,72,35,18,98 |

Output

11

**Explanation**

The base 6 representation of this is 320,33,115,144,200,55,30,242, and the derived sequence (sum of digits) is 5,6,7,9,2,10,3,8. The number of inversions is 11 (5,2), (5,3),(6,2) (6,3), (7,2), (7,3) (9,2),(9,3) (9,8),(10,3), (10,8)

#include<stdio.h>

int i;

int rem\_val;

// A utility function to print an array p[] of size 'n'

void printArray(int p[], int n)

{

for (i = 0; i < n; i++)

{

printf("%d",p[i]);

}

printf("\n");

}

void printAllUniqueParts(int n)

{

int p[100]; // An array to store a partition

int k = 0; // Index of last element in a partition

p[k] = n; // Initialize first partition as number itself

// This loop first prints current partition, then generates next

// partition. The loop stops when the current partition has all 1s

while (1)

{

// print current partition

printArray(p, k+1);

// Generate next partition

// Find the rightmost non-one value in p[]. Also, update the

// rem\_val so that we know how much value can be accommodated

rem\_val = 0;

while (k >= 0 && p[k] == 1)

{

rem\_val += p[k];

k--;

}

// if k < 0, all the values are 1 so there are no more partitions

if (k < 0) return;

// Decrease the p[k] found above and adjust the rem\_val

p[k]--;

rem\_val++;

// If rem\_val is more, then the sorted order is violated. Divide

// rem\_val in different values of size p[k] and copy these values at

// different positions after p[k]

while (rem\_val > p[k])

{

p[k+1] = p[k];

rem\_val = rem\_val - p[k];

k++;

}

// Copy rem\_val to next position and increment position

p[k+1] = rem\_val;

k++;

}

}

// Driver program to test above functions

main()

{

printf( "All Unique Partitions of 2 \n");

printAllUniqueParts(2);

printf("\nAll Unique Partitions of 3 \n");

printAllUniqueParts(3);

printf("\nAll Unique Partitions of 4 \n");

printAllUniqueParts(4);

getch();

}

2)

## Digital Time

### Problem Description

The objective is to form the maximum possible time in the HH:MM:SS format using any six of nine given single digits (not necessarily distinct)

Given a set of nine single (not necessarily distinct) digits, say 0, 0, 1, 3, 4, 6, 7, 8, 9, it is possible to form many distinct times in a 24 hour time format HH:MM:SS, such as 17:36:40 or 10:30:41 by using each of the digits only once. The objective is to find the maximum possible valid time (00:00:01 to 24:00:00) that can be formed using some six of the nine digits exactly once. In this case, it is 19:48:37.

### Input Format

A line consisting of a sequence of 9 (not necessarily distinct) single digits (any of 0-9) separated by commas. The sequence will be non-decreasing

### Output

The maximum possible time in a 24 hour clock (00:00:01 to 24:00:00) in a HH:MM:SS form that can be formed by using some six of the nine given digits (in any order) precisely once each. If no combination of any six digits will form a valid time, the output should be the word Impossible

### Explanation

**Example 1**

Input

0,0,1,1,3,5,6,7,7

Output

17:57:36

The maximum valid time in a 24 hour clock that can be formed using some six of the 9 digits precisely once is 17:57:36

**Example 2**

Input

3,3,3,3,3,3,3,3,3

Output

Impossible

No set of six digits from the input may be used to form a valid time.

#include<iostream.h>

#include<conio.h>

//Applying the concept of count Array

int count[10]={0};

//This function will return the maximum value from count array upto

//the given index n

int MAX(int n)

{

int i;

for(i=n;i>=0;i--)

{

if(count[i]!=0)

{

count[i]--;

return i;

}

}

return -1;

}

//main program

main()

{

int x,i,y=0;

char A[8];

for(i=0;i<9;i++)

{

cin>>x;

if(x>=0 && x<=9)

count[x]++;

else

{

cout<<"Wrong Input!Please Enter Again"<<endl;

i--;

}

}

if(count[2]>=1 && count[1]>=1 && count[0]>=4)

cout<<"12:00:00"<<endl;

else

{

//All works for char array A[]

for(i=0;i<8 && y!=1;i++)

{

if(i%3==2)

{

A[i]=':';

}

else if(i%3==0 && i>0)

{

//Adding '0' to convert the value into its character format

A[i]=MAX(5)+'0';

}

else if(i%3==1)

{

if(A[0]==0+'0' || i>1)

{

A[i]=MAX(9)+'0';

}

else

{

A[i]=MAX(1)+'0';

}

}

else if(i==0)

A[i]=MAX(1)+'0';

if(A[i]==-1+'0')

y=1;

}

if(y==1)

{

cout<<"Impossible Operation!"<<endl;

}

else

{

for(i=0;i<8;i++)

if(i%3==2)

cout<<A[i];

else

cout<<A[i]-'0';

cout<<endl;

}

}

getch();

}

## c) Counting Rock samples

### Problem Description

Juan Marquinho is a geologist and he needs to count rock samples in order to send it to 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 the range of the laboratory. This process is very hard because the rock samples may be in millions.

Juan Marquinho needs your help, your task is develop a program to get the number of rocks of a given range of size.

### Constraints

10 <= S <= 10000

1 <= R <= 1000000

1<=size of Sample <= 1000

### 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 ith line containing two positive integers, space separated, indicating the minimum size and maximum size respectively of the ith range.

### Output

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

### Explanation

**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 sampes (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.

#include<stdio.h>

#include<conio.h>

main()

{

int n[100],m[100],k[100],i,t,c[100],b;

clrscr();

scanf("%d",&t);

for(i=0;i<t;i++)

{

scanf("%d%d%d",&n[i],&k[i],&m[i]);

}

for(i=0;i<t;i++)

{

c[i]=1;

n[i]=n[i]+k[i];

b=m[i];

while(n[i]!=m[i])

{

if(n[i]>m[i])

{

m[i]+=b;

c[i]++;

}

else

{

n[i]=n[i]+k[i];

}

}

}

for(i=0;i<t;i++)

{

printf("%d\n",c[i]);

}

getch();

}

## d) Great Chase

### Problem Description

Welcome to the Planet **Zandar**, the second most prominent planet in the Milky Way Galaxy (of course after our own Earth).

The planet  is in a distress condition, a Group of Galactic pirates, **Zorons**have stolen the **Trident Crystal**, which is the main source of energy of the planet, and are escaping the Galaxy. **The Nova Corps,** the military agency of Zandar, have gathered intelligence that the Zoronion space craft can run in cosmic leaps of exactly D units, (it means that the space craft will move D units from its position in every leap/turn) and is currently I units away from Zandar.

The Zandarian Space crafts can run in cosmic leaps of exactly Z units. The Commander of Nova Corps wants to know the smallest number of leaps required to catch Zorons (Note that it is possible to catch the pirates only when they will be at the same point in the cosmic universe). The Zorons, even though are clever thieves, travel in one direction, and keep jumping exactly D units without stopping at any point.

The Nova Corps can dial in the number of jumps they need to make (each of them exactly Z units), and reach the place almost instantly. They can then wait there until the Zorons arrive, and recover the Trident Crystal.

However, their wizard has told them that there may be situations where it is impossible for the Nova corps to be at the same distance as the Zorons.

As the planet is out of power currently, their supercomputers are shut down and they are not able to calculate the required information. As you are there from Earth they have approached you for help.

**Note: Assume that the Cosmic universe is one dimensional.**

### Constraints

1 <= I,D <= 10^12

1<= Z <= 10^9

### Input Format

An integer T for number of test cases, followed by T test cases each one consisting of three numbers

1) I :- initial distance of Zorons

2) D:- distance covered in a single cosmic leap by Zoronion space craft.

3) Z:- distance covered by Zandarian space crafts.

### Output

Single number, the number of leaps required to catch the pirates, and if it is not possible to catch them, output will be -1

### Explanation

**Example 1**

Input

2 9 5 12 5 7 9

Output

2 6

Explanation

The first line is 2, so T is 2, and there are 2 test cases.

In the first test case, The Zorons will initially be at 9 and then they will leap to 14,19 24.....  The Nova Corps can take leaps of 12 and will catch them nearest at a distance 24, taking 2 leaps 12 and 24.

In the second test case, The Zorons will initially be at 5 and then they will leap to 12 19 26 33, 40, 47, 54.....  The Nova Corps can take leaps of 9 and will catch them nearest at 54, taking 6 leaps.

Hence the output has 2 lines, 2 and 6.

**Example 2**

Input

1

10 15 20

Output

2

Explanation

The first line is 1, so T is 1, and there is 1 test case.

The Zorons will initially be at 10, and jump in jumps of 15, landing at 25,40,

The Nova Corps take leaps of 20, and arrive at 20, 40. Hence, they can meet at 40 after 2 leaps. The output is 2.

## Distinct Partition Squares

### Problem Description

Among the several path breaking contributions to Number theory by the famous Indian mathematician Srinivasa Ramanujan, his contribution to partitions is extensive and deep. A partition of a positive integer n, also called an integer partition, is a way of writing n as a sum of positive integers. Two sums that differ only in the order of their summands are considered the same partition. For example, 4 can be expressed as a sum of positive integers in the following ways: 1+1+1+1, 1+1+2, 1+3, 2+2, 4. Of these, only 1+3 and 4 use non repeating summands. Partitions using non repeating summands are called distinct partitions of n. There is no general formula for the number of partitions of an integer n and it is known that the partitions grow rapidly with n.

A k-distinct-partition of a number n is a set of k distinct positive integers that add up to n. For example, 3-distinct partitions of 10 are 1+2+7, 1+3+6,1+4+5 and 2+3+5

The objective is to count all k-distinct partitions of a number that have at least two perfect squares in the elements of the partition. Note that 1 is considered a perfect square.

### Constraints

k<N<200, so that at least one k-distinct partition exists.

### Input Format

The input consists of one line containing of N and k separated by a comma.

### Output

One number denoting the number of k-distinct partitions of N that have at least two perfect squares in the elements of the partition.

### Explanation

**Example 1**

Input

10, 3

Output

1

Explanation: The input asks for 3-distinct-partitions of 10. There are 4 of them (1+2+7, 1+3+6, 1+4+5 and 2+3+5). Of these, only 1 has at least two perfect squares in the partition (1+4+5).

**Example 2**

Input

12, 3

Output

2

Explanation

The input asks for 3-distinct partitions of 12. There are 7 of them (9+2+1, 8+3+1,7+4+1,7+3+2,6+5+1, 6+4+2, 5+4+3). Of these, two, (9+4+1, 7+4+1) have two perfect squares. Hence, the output is 2.

## Distint Partitions

### Problem Description

Among the several path breaking contributions to Number theory by the famous Indian mathematician Srinivasa Ramanujan, his contribution to partitions is extensive and deep. A partition of a positive integer n, also called an integer partition, is a way of writing n as a sum of positive integers. Two sums that differ only in the order of their summands are considered the same partition. For example, 4 can be expressed as a sum of positive integers in the following ways,: 1+1+1+1, 1+1+2, 1+3, 2+2, 4. Of these, only 1+3 and 4 use non repeating summands. Partitions using non repeating summands are called distinct partitions of n. There is no general formula for the number of partitions of an integer n and it is known that the partitions grow rapidly with n.

A k-distinct-partition of a number n is a set of k distinct positive integers that add up to n. Hence, if we look at 3-distict partitions of 10, they are the partitions 1+2+7, 1+3+6,1+4+5 and 2+3+5

The objective is to count all k-distinct partitions of a number that have at least two prime numbers in the elements of the partition

### Constraints

k<N<200, so that at least one k-distinct partition exists

### Input Format

The input consists of one line containing of N and k separated by a comma

### Output

One number denoting the number of k-distinct partitions of N that have at least two prime numbers in the elements of the partition.

### Explanation

**Example 1**

Input

10,3

Output

2

Explanation : The input asks for 3-distinct-partitions of 10. There are 4 of them (1+2+7, 1+3+6, 1+4+5 and 2+3+5). Of these, only 2 have at least two primes in the partition (1+2+7, 2+3+5)

**Example 2**

Input

12,3

Output

2

Explanation : The input asks for 3-distinct partitions of 12. There are 7 of them (1+2+9, 1+3+8, 1+4+7, 2+3+7, 1+5+6, 2+4+6 , 3+4+5). Of these 2 (2+3+7 and 3+4+5) have at least 2 primes. Hence the output is 2.