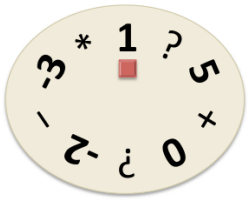
The Safe Secret
Problem ID: safesecret



One of the brightest and richest dukes of the nineteenth century built a break-in-proof room for storing his valuables and chose the lock secret code in an ingenious manner. He was so afraid of being robbed that he did not tell anyone the safe secret; he only wrote the way to obtain it on a piece of paper, to be given to his heir on his death.

1. Look at the bottom of my dukedom ring, which is now yours.
2. Write down the numbers and symbols, following a clockwise order, starting at the number closest to the ruby and leaving out the last symbol. That is the first sequence of numbers and symbols.
   Do the same starting at the next number, with respect to the clockwise order. That is the second sequence of numbers and symbols.
   Repeat this process, always starting at the next number, until you have started at all numbers. Now you have several sequences of numbers and symbols.
3. For each of those sequences of numbers and symbols, do the following.
   (a) Replace every ? by a $+$, a $-$ or a $*$ symbol.
   Do that in all possible ways to have several arithmetic expressions.
   (b) Evaluate each of those arithmetic expressions, performing the sums, the differences and the products in any order.
   Do that in all possible ways to have several values.
   (c) Select the minimum and the maximum of those values.
   (d) Write the digits of the minimum value and append to them the digits of the maximum value. That is the code of the sequence of numbers and symbols.
4. Concatenate the codes of all sequences of numbers and symbols, respecting the order in which you have obtained the sequences. That sequence of digits is the safe secret.

When the duke passed away, his son read the note and tried to find out the safe secret. The first two steps were very easy, because there were only five sequences of numbers and symbols, obtained in the following order:

1	?	5	+	0	?	-2	−	-3
5	+	0	?	-2	−	-3	∗	1
0	?	-2	−	-3	∗	1	?	5
-2	−	-3	∗	1	?	5	+	0
-3	∗	1	?	5	+	0	?	-2

Then, he moved to the third step and chose to begin with the first sequence of numbers and symbols. Difficulties started in point (a) when he realised that he could create several arithmetic expressions, such as:

$$1 + 5 + 0 + \text{-2} - \text{-3}, \quad 1 - 5 + 0 * \text{-2} - \text{-3}, \quad \text{and} \quad 1 * 5 + 0 - \text{-2} - \text{-3}.$$

So, he decided to understand the remaining rules before completing this task. In point (b), he had to evaluate the arithmetic expressions. It seemed easy. The value of $1 + 5 + 0 + \text{-2} - \text{-3}$ was **7**. But how many different values could he get from $1 - 5 + 0 * \text{-2} - \text{-3}$?

- If the operations were performed from left to right, $((((1 - 5) + 0) * \text{-2}) - \text{-3})$, the result would be **11**.
- If the operations were performed from right to left, $(1 - (5 + (0 * (\text{-2} - \text{-3}))))$, the result would be **-4**.

- If the first difference and the product were performed first, $(1 - 5) + (0 * \text{-}2) - \text{-}3$, the result would be **-1**.
- And there were so many other alternatives!

Almost in despair, he concluded that he had to obtain a huge number of values in the third step. Fortunately, the last rules were actually simple. If **-4** was the minimum of the values obtained from the first sequence and **11** was the maximum, the code of the first sequence would be **411**. Besides, if the second sequence code was **512**, the third sequence code was **613**, the fourth sequence code was **714**, and the fifth sequence code was **815**, the safe secret would be **411512613714815**.

Although the duke's son spared no effort in finding the secret, he has never achieved that goal. In fact, no one has managed to open the safe so far. Now that the palace will be transformed into a museum, could you help unveiling the treasure?

Task

Given the sequence of numbers and symbols obtained from the dukedom ring, starting at the number closest to the ruby, following a clockwise order, and including the last symbol, the goal is to find out the safe secret. It is guaranteed that, for the given inputs, any value obtained by the process described above fits in a normal signed 64 bit integer.

Input

The first line of the input has one positive integer, k, which is the number of pairs (*number*, *symbol*) that form the sequence.

The following line contains $2k$ elements, $n_1, s_1, n_2, s_2, \ldots, n_k, s_k$, separated by a single space, where n_i denotes a number and s_i denotes a symbol that is $+$, $-$, $*$, or ? (for every $i = 1, 2, \ldots, k$).

Constraints

$2 \leq k \leq 200$	Number of pairs (*number*, *symbol*) that form the sequence.
$-9 \leq n_i \leq 9$	Number in the sequence.

Output

The output has a single line with the safe secret.

Sample Input 1	Sample Output 1
5 1 ? 5 + 0 ? -2 - -3 *	914710203014163336

Stacking Curvy Blocks

Problem ID: curvyblocks

You're doing some construction work, and, to save money, you're using some discount, "irregular" construction materials. In particular, you have some blocks that are mostly rectangular, but with one edge that's curvy. As illustrated below, you're going to use these irregular blocks between stacks of ordinary blocks, so they won't shift sideways or rotate. You'll put one irregular block on the bottom, with its curvy edge pointing up, and another above it, with it's curvy edge pointing down. You just need to know how well these blocks fit together. You define the fit quality as the maximum vertical gap between the upper edge of the bottom block and the lower edge of the top block when the upper block is just touching the lower one.

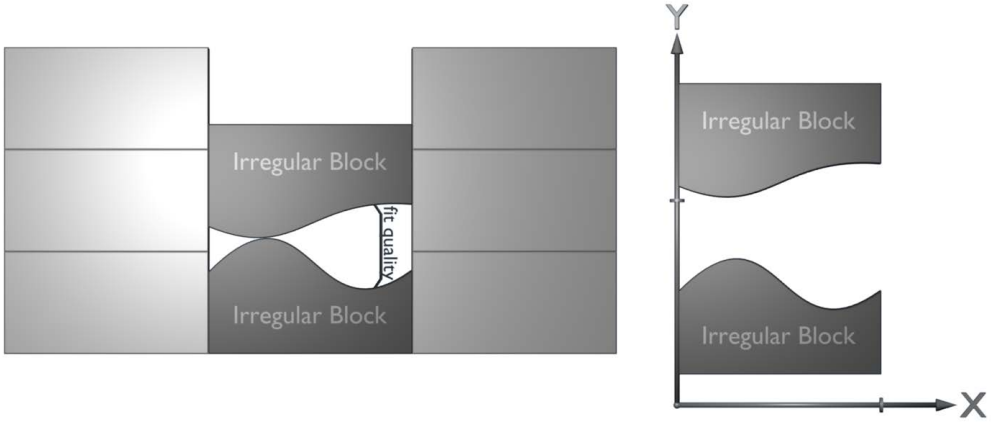


Figure 1: Block stacking and coordinate system

All blocks are one unit wide. You've modeled the curvy edges as cubic polynomials, with the left edge of the block at $x = 0$ and the right edge at $x = 1$.

Input

Each test case is given on two lines, with each line containing four real numbers. The numbers on the first line, $b_0\ b_1\ b_2\ b_3$, describe the shape of the upper edge of the bottom block. This edge is shaped just like the polynomial $b_0 + b_1x + b_2x^2 + b_3x^3$ for $0 \leq x \leq 1$. The numbers on the next input line, $t_0\ t_1\ t_2\ t_3$, describe the bottom edge of the block that's going on top. This edge is shaped just like the polynomial $t_0 + t_1x + t_2x^2 + t_3x^3$ for $0 \leq x \leq 1$. No input value will have magnitude greater than 50000. There are at most 500 test cases. Input ends at end of file.

Output

For each test case, print out a single line giving the fit quality. An answer is considered correct if its absolute or relative error is at most 10^{-7}.

Sample Input 1	Sample Output 1
1.000000 -12.904762 40.476190 -28.571429 3.000000 11.607143 -34.424603 22.817460 2.000000 -10.845238 16.964286 -10.119048 3.000000 4.190476 -3.571429 2.380952	4.396074 6.999999

Paintball
Problem ID: paintball

Marek and his schoolmates have just finished their studies at the university. They wanted to celebrate it with a game of paintball. After an hour of playing a very strange thing happened – everyone had exactly one bullet left. Marek, being a very curious person, wanted to know whether it's possible that everyone will be hit exactly once provided nobody moves.

Task

You are given a description of the situation during a paintball game when every player has only one bullet. The description of the game consists of pairs of players who can see each other. If a player can see another player, he can fire at him. Your task is to find a target for each player such that everyone will be hit.

Input

The first line of input contains two space separated integers N and M, satisfying $2 \leq N \leq 1\,000$ and $0 \leq M \leq 5\,000$, where N is the number of players. Players are numbered $1, 2, \ldots, N$. M lines follow, each line containing two space separated integers A and B ($1 \leq A < B \leq N$), denoting that players A and B can see each other. Each pair of players appears at most once in the input.

Output

If there is no assignment of targets such that everyone will be hit, output `Impossible`. Otherwise output N lines. The i-th line should contain the number of the target of the i-th player. If there is more than one solution, output any one.

Sample Input 1

```
3 3
1 2
2 3
1 3
```

Sample Output 1

```
2
3
1
```

Sample Input 2

```
3 2
1 2
1 3
```

Sample Output 2

```
Impossible
```

Guessing Camels

Problem ID: camels

Time limit: 10 seconds

Jaap, Jan, and Thijs are on a trip to the desert after having attended the ACM ICPC World Finals 2015 in Morocco. The trip included a camel ride, and after returning from the ride, their guide invited them to a big camel race in the evening. The camels they rode will also participate and it is customary to bet on the results of the race.

One of the most interesting bets involves guessing the complete order in which the camels will finish the race. This bet offers the biggest return on your money, since it is also the one that is the hardest to get right.

Jaap, Jan, and Thijs have already placed their bets, but the race will not start until an hour from now, so they are getting bored. They started wondering how many pairs of camels they have put in the same order. If camel c is before camel d on Jaap's, Jan's and Thijs' bet, it means that all three of them put c and d in the same order. Can you help them to calculate the number of pairs of camels for which this happened?

Jaap, Jan and Thijs on camels (in some order). Photo by Tobias Werth, cc by-sa.

Input

The input consists of:

- one line with an integer n ($2 \leq n \leq 200\,000$), the number of camels;
- one line with n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq n$ for all i), Jaap's bet. Here a_1 is the camel in the first position of Jaap's bet, a_2 is the camel in the second position, and so on;
- one line with Jan's bet, in the same format as Jaap's bet;
- one line with Thijs' bet, in the same format as Jaap's bet.

The camels are numbered $1, \ldots, n$. Each camel appears exactly once in each bet.

Output

Output the number of pairs of camels that appear in the same order in all 3 bets.

Sample Input 1

```
3
3 2 1
1 2 3
1 2 3
```

Sample Output 1

```
0
```

Sample Input 2

```
4
2 3 1 4
2 1 4 3
2 4 3 1
```

Sample Output 2

```
3
```

Server

Problem ID: server

Time limit: 1 second

You are in charge of a server that needs to run some submitted tasks on a first-come, first-served basis. Each day, you can dedicate the server to run these tasks for at most T minutes. Given the time each task takes, you want to know how many of them will be finished today.

Consider the following example. Assume $T = 180$ and the tasks take $45, 30, 55, 20, 80$, and 20 minutes (in order they are submitted). Then, only four tasks can be completed. The first four tasks can be completed because they take 150 minutes, but not the first five, because they take 230 minutes which is greater than 180. Notice that although there is enough time to perform the sixth task (which takes 20 minutes) after completing the fourth task, you cannot do that because the fifth task is not done yet.

Picture from Wikimedia Commons

Input

The input consists of a single test case. The first line contains two integers n and T where $1 \leq n \leq 50$ is the number of tasks and $1 \leq T \leq 500$. The next line contains n positive integers no more than 100 indicating how long each task takes in order they are submitted.

Output

Display the number of tasks that can be completed in T minutes on a first-come, first-served basis.

Sample Input 1	Sample Output 1
6 180 45 30 55 20 80 20	4

Sample Input 2	Sample Output 2
10 60 20 7 10 8 10 27 2 3 10 5	5

Shuffling Along

Problem ID: shuffling

Most of you have played card games (and if you haven't, why not???) in which the deck of cards is randomized by shuffling it one or more times.

A *perfect shuffle* is a type of shuffle where the initial deck is divided exactly in half, and the two halves are perfectly interleaved. For example, a deck consisting of eight cards ABCDEFGH (where A is the top card of the deck) would be divided into two halves ABCD and EFGH and then interleaved to get AEBFCGDH. Note that in this shuffle the original top card (A) stays on top — this type of perfect shuffle is called an *out-shuffle*. An equally valid perfect shuffle would start with the first card from the second half and result in EAFBGCHD — this is known as an *in-shuffle*.

While normal shuffling does a good job at randomizing a deck, perfect shuffles result in only a small number of possible orderings. For example, if we perform multiple out-shuffles on the deck above, we obtain the following:

$$\text{ABCDEFGH} \rightarrow \text{AEBFCGDH} \rightarrow \text{ACEGBDFH} \rightarrow \text{ABCDEFGH} \rightarrow \cdots$$

So after 3 out-shuffles, the deck is returned to its original state. A similar thing happens if we perform multiple in-shuffles on an 8-card deck, though in this case it would take 6 shuffles before we get back to where we started. With a standard 52 card deck, only 8 out-shuffles are needed before the deck is returned to its original order (talented magicians can make use of this result in many of their tricks). These shuffles can also be used on decks with an odd number of cards, but we have to be a little careful: for out-shuffles, the first half of the deck must have 1 more card than the second half; for in-shuffles, it's the exact opposite. For example, an out-shuffle on the deck ABCDE results in ADBEC, while an in-shuffle results in CADBE.

For this problem you will be given the size of a deck and must determine how many in- or out-shuffles it takes to return the deck to its pre-shuffled order.

Input

The input consists of one line containing a positive integer $n \leq 1000$ (the size of the deck) followed by either the word `in` or `out`, indicating whether you should perform in-shuffles or out-shuffles.

Output

For each test case, output the case number followed by the number of in- or out-shuffles required to return the deck to its original order.

Sample Input 1	Sample Output 1
`8 out`	`3`

Sample Input 2	Sample Output 2
`8 in`	`6`

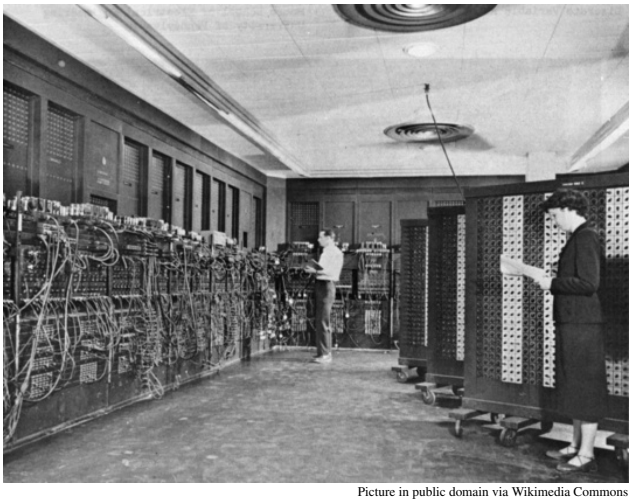
Sample Input 3	Sample Output 3
`52 out`	`8`

Sample Input 4	Sample Output 4
`53 out`	`52`

Primary Register

Problem ID: register

We're working on a new super-computer, built to unleash the hidden computing powers of all eight dimensions of reality. The secret to utilizing more dimensions turns out to be to use counting registers of different sizes. In particular, we have eight registers, counting cyclically modulo the different primes 2, 3, 5, 7, 11, 13, 17, 19. A size p register stores a value between 0 and $p-1$ (inclusive).



Picture in public domain via Wikimedia Commons

The only operation available is an "increment" operation. When it is performed, the size 2 register is increased by 1. If this increment causes overflow (i.e., if the old value was 1) the value is reset to 0, and the size 3 is incremented. If this causes overflow the size 3 register is reset to 0 and the size 5 register is incremented, and so on. If this goes all the way to the last register and the size 19 register overflows, the computer blows up.

In order not to destroy the computer in testing, we need to build a program to check the safety of doing increment operations before we perform them. Given the current state of the registers, you need to compute how many more operations can safely be performed before the computer blows up.

Input

The input consists of a single line containing eight integers v_2, v_3, v_5, v_7, v_{11}, v_{13}, v_{17}, v_{19} indicating the current values of the registers. The value of the size p register is always between 0 and $p-1$ (inclusive).

Output

Ouput a single line containing an integer N, the number of additional operations that can be performed without the computer blowing up.

Sample Input 1	Sample Output 1
0 0 4 6 10 12 16 18	5

Sample Input 2	Sample Output 2
1 2 4 6 10 12 16 18	0

Sample Input 3	Sample Output 3
0 0 0 0 0 0 0 0	9699689

Grandpa Bernie

Problem ID: grandpabernie

Over the years, Grandpa Bernie has traveled all over the world. He doesn't travel that much anymore, but he loves to tell his grandchildren stories from all these trips. He'll tell them the story from when he went to Israel for the first time, or when he went to Greece for the third time.

His memory works in a funny way. He can easily remember his k:th trip to a particular country, but he'll have a hard time remembering in which year he went on that trip. Given a list of all the trips Grandpa Bernie went on, can you answer a number of queries asking in which year he went on his k:th trip to a particular country?

Input

The input consists of:

1. one line with one integer n ($1 \leq n \leq 10^5$), the number of trips Grandpa Bernie went on;
2. n lines each containing the name s ($1 \leq |s| \leq 20$) of a country and an integer y ($1 \leq y \leq 10^6$) representing a trip to country s that Grandpa Bernie went on in year y;
3. one line with one integer q ($1 \leq q \leq 10^5$), the number of queries;
4. q lines each containing the name s of a country and an integer k representing a query for the k:th time Grandpa Bernie went to country s.

Each country name only consists of letters from the English alphabet. It is also guaranteed that, for each query asking for the k:th trip to country s, k is at least 1 and no greater than the number of times Grandpa Bernie went to country s. In particular, it is guaranteed that Grandpa Bernie has visited country s at least once.

Output

For each query for the k:th trip Grandpa Bernie went to a country s, output a single line containing the year in which Grandpa Bernie went on that trip.

Sample Input 1

```
4
Iceland 2016
Sweden 2015
Iceland 1982
Norway 1999
3
Sweden 1
Iceland 1
Iceland 2
```

Sample Output 1

```
2015
1982
2016
```

Sample Input 2

```
4
Iceland 2014
Iceland 2015
Iceland 2015
Iceland 2016
4
Iceland 4
Iceland 3
Iceland 2
Iceland 1
```

Sample Output 2

```
2016
2015
2015
2014
```

Pub-lic Good
Problem ID: pubs

A bit over three years ago, you were elected president of the glorious and picturesque country of Molvanîa, a land untouched by modern dentistry. To secure your landslide victory in the election you had to make a few promises, some of which, with the clarity of hindsight, may have been a tad exaggerated.

Molvanîa's economy is quite simple compared to that of most other countries. The two main professions in Molvanîa are pub-owners, and beer-drinkers. These two groups combined account for over 75% of the Molvanîan GDP (Gross Domestic Product). Slightly simplified, the system works like this: the beer-drinkers borrow money to pay for their beer. This creates income for the pub-owners. The pub-owners use their income to purchase AAA-rated bonds, backed by loans to beer-drinkers. This system is locally referred to as pub-prime lending.

Photo by National Library of Australia

Task

One of your election-time promises was to further optimize Molvanîa's Tiger economy through improved city planning. You have identified a number of suitable construction sites in which either a pub or a house of a beer-drinker can be built. There are walkways between some of these sites. To fully optimize the economy, you want to place buildings such that each house has at least one pub at only a walkway's distance, and each pub has at least one house at only a walkway's distance. It might happen that this is impossible, but you will try your best.

Beware that the city has a quite peculiar lay-out, and it may not even be possible to draw it on a normal map. Molvanîa is special that way.

Input

There are n construction sites and m walkways in the city ($1 \leq n \leq 10\,000$ and $0 \leq m \leq 100\,000$). The first line contains n and m, separated by a single space. The next m lines contain integers x and y ($1 \leq x, y \leq n$) indicating that there is a walkway between x and y. There are no loops (i.e., $x \neq y$) and all lines with walkway descriptions are distinct.

Output

If it is impossible to build pubs and houses such that every pub is next to a house and every house is next to a pub, print `Impossible` on a line. Otherwise output n space separated words. Print `pub` or `house` for each construction site. The first word indicates what to build at construction site 1, the next at construction site 2, and so on. If there are multiple valid solutions, you can output any of them.

Sample Input 1	Sample Output 1
4 4 1 2 2 3 3 4 4 1	pub house pub house

Sample Input 2	Sample Output 2
4 4 1 2 2 3 3 4 4 2	pub house pub house

Sunlight

Problem ID: sunlight

A core right in Roman tenancy law was the availability of sunlight to everybody, regardless of status. Good sun exposure has a number of health benefits, many of which were known even in those ancient times.

The first act of a Roman city plan reviewer, then, is to survey the proposed structures to measure how well they distribute this precious resource. Given any one avenue of buildings arranged West-to-East, the number of hours for which each building is exposed to sunlight needs to be determined.

For the purpose of simplicity, the number of hours a building is in sunlight is proportional to the fraction of the 180 degrees of sky visible from its top. Thanks in no small part to the marvels of ancient engineering (and also to the strict nutritional regimes of old) you may assume each building is infinitesimally thin.

Input

- One line containing one integer N ($1 \leq N \leq 2 \cdot 10^5$): the number of buildings.
- N further lines each containing two space-separated integers X_i and H_i ($1 \leq X, H \leq 10^9$), the location and height respectively, in metres, of the i^{th}-most building from the west.

Output

On each of the N lines of output, write one real number to at least 4 decimal places of accuracy: the number of hours for which the peak of the i-th building is bathed in sunlight.

Sample Input 1

```
4
1 1
2 2
3 2
4 1
```

Sample Output 1

```
9.0000
12
12.00000
9.0
```

Sample Input 2

```
5
100 50
125 75
150 100
175 125
200 25
```

Sample Output 2

```
9.0
9.0
9.0
12.0
6.9357496
```

Bank Queue
Problem ID: bank

Oliver is a manager of a bank near KTH and wants to close soon. There are many people standing in the queue wanting to put cash into their accounts after they heard that the bank increased the interest rates by 42% (from 0.01% per year to 0.0142% per year).

However, there are too many people and only one counter is open which can serve one person per minute. Greedy as Oliver is, he would like to select some people in the queue, so that the total amount of cash stored by these people is as big as possible and that money then can work for the bank overnight.

There is a problem, though. Some people don't have the time to wait until the bank closes because they have to run somewhere else, so they have to be served before a certain time, after which they just leave. Oliver also turned off the infrared door sensor outside the bank, so that no more people can enter, because it's already too crowded in the hall.

Task

Help Oliver calculate how much cash he can get from the people currently standing in the queue before the bank closes by serving at most one person per minute.

Input

The first line of input contains two integers N ($1 \leq N \leq 10\,000$) and T ($1 \leq T \leq 47$), the number of people in the queue and the time in minutes until Oliver closes the bank. Then follow N lines, each with 2 integers c_i and t_i, denoting the amount of cash in Swedish crowns person i has and the time in minutes from now after which person i leaves if not served. Note that it takes one minute to serve a person and you must begin serving a person at time t_i at the latest. You can assume that $1 \leq c_i \leq 100\,000$ and $0 \leq t_i < T$.

Output

Output one line with the maximum amount of money you can get from the people in the queue before the bank closes.

Sample Input 1

```
4 4
1000 1
2000 2
500 2
1200 0
```

Sample Output 1

```
4200
```

Sample Input 2

```
3 4
1000 0
2000 1
500 1
```

Sample Output 2

```
3000
```

Saving For Retirement

Problem ID: savingforretirement

Alice is saving for her retirement. She hasn't really decided how much she wants to save, but when she retires, she wants to have strictly more money than Bob will have when he retires.

Bob is B years old. He plans to retire when he becomes B_r years old. He saves B_s every year from now until then.

Alice is A years old. She wants to save A_s every year. When is the earliest time she can retire?

Input

The input is a single line consisting of 5 space separated integers; B, B_r, B_s, A, A_s.

Output

Output the age at which Alice can retire so that she has more money than Bob will have at age B_r.

Limits

- $20 \leq B \leq B_r \leq 100$
- $20 \leq A \leq 100$
- $1 \leq A_s, B_s \leq 100$

Explanation of first example

At the age of 25 Bob has saved 5 every year for 5 years. This means he has 25 saved up.
At the age of 23 Alice has saved 10 every year for 3 years. This means she has 30 saved up, which is strictly more than 25.

Sample Input 1	Sample Output 1
20 25 5 20 10	23

Sample Input 2	Sample Output 2
20 28 5 30 9	35