

New Zealand Programming Contest 2019

# PROBLEM SET

## PREAMBLE

Please note the following very important details relating to input and output:

- Read all input from the keyboard, i.e. use `stdin`, `System.in`, `cin` or equivalent. Input will be redirected from a file to form the input to your submission.
- Do NOT prompt for input as this will appear in your output and cause a submission to be judged as wrong.
- Write all output to the screen, i.e. use `stdout`, `System.out`, `cout` or equivalent. Do not write to `stderr`. Do NOT use, or even include, any module that allows direct manipulation of the screen, such as `conio`, `Crt` or anything similar.
- Output from your program is redirected to a file for later checking. Use of direct I/O means that such output is not redirected and hence cannot be checked. This could mean that a correct program is rejected! You have been warned.
- Unless otherwise stated, all *integers* will fit into a standard 32-bit computer word. If more than one integer appears on a line, they will be separated by spaces.
- An *uppercase letter* is a character in the sequence 'A' to 'Z'. A *lower case letter* is a character in the sequence 'a' to 'z'. A *letter* is either a lower case letter or an upper case letter.
- Unless otherwise stated, a *word* or a *name* is a continuous sequence of letters.
- Unless otherwise stated, a *string* is a continuous sequence of visible characters.
- Unless otherwise stated, words and names will contain no more than 60 characters, and strings will contain no more than 250 characters.
- If it is stated that 'a line contains no more than  $n$  characters', this does not include the character(s) specifying the end of line.
- Input files are often terminated by a 'sentinel' line, followed by an end of file marker. This line should not be processed.

**PROBLEM A**

**CAR PARKING**

**3 POINTS**

The manager of a car park has asked for your help with an occupancy survey. He wants to know how many spaces in the car park are occupied on consecutive days.

**Input**

The first line in the input will be a single positive integer,  $N$  ( $0 < N \leq 250$ ), which is the number of parking spaces available.

The second line contains data for the first day. It consists of  $N$  characters,  $O$  for an occupied space, with  $-$  for an empty space.

The third line contains similar data for the second day.

**Output**

Output consists of a single integer, the number of parking spaces that were occupied on both days.

**Sample Input 1**

5  
00-00  
-00--

**Output for Sample Input 1**

1

**Explanation**

Only parking space 2 (numbering from the left) was occupied on both days.

**Sample Input 2**

7  
0000000  
00-0--0

**Output for Sample Input 2**

4

**Explanation**

Parking spaces 1, 2, 4 and 7 were occupied on both days.



**PROBLEM B**

**ARITHMETIC**

**3 POINTS**

How is your arithmetic? In this problem, you will be presented with 2 numbers and have to calculate and display their sum, difference, product, quotient and remainder.

**Input**

Input will consist of 2 numbers, N1 and N2, each on a line of its own. Both numbers will be integers between -100 and 100 inclusive. N2 will not be 0.

**Output**

Output 4 lines showing the required values. The required output format is:

N1 + N2 = <sum>

N1 - N2 = <difference>

N1 x N2 = <product>

N1 divided by N2 is <quotient> remainder <remainder>

where <sum> is  $N1 + N2$ , <difference> is  $N1 - N2$ , <product> is  $N1 \times N2$ , <quotient> is N1 divided by N2 (integer part only) and <remainder> is the remainder when N1 is divided by N2.

**Turn over for sample input and output**

### Sample Input 1

27  
14

### Output for Sample Input 1

$27 + 14 = 41$   
 $27 - 14 = 13$   
 $27 \times 14 = 378$   
27 divided by 14 is 1 remainder 13

### Sample Input 2

-8  
11

### Output for Sample Input 2

$-8 + 11 = 3$   
 $-8 - 11 = -19$   
 $-8 \times 11 = -88$   
-8 divided by 11 is 0 remainder -8

### Sample Input 3

15  
-7

### Output for Sample Input 3

$15 + -7 = 8$   
 $15 - -7 = 22$   
 $15 \times -7 = -105$   
15 divided by -7 is -2 remainder 1

**PROBLEM C****COOKING****3 POINTS**

Most people use a recipe when cooking something for the first time. They also check that they have all the ingredients they need before they start! This problem will require you to help with that process.

**Input**

The first line of input will contain the name of the item being made. It will be no more than 20 characters which may include spaces.

The next line will contain a single positive integer,  $N$ , the number of ingredients required ( $N \leq 20$ ).

The next  $N$  lines will each contain details of 1 ingredient. The format for this will be

`<name>`, `<units>`, `<amount>`

`<name>` will be the name of the ingredient with up to 20 lower case letters or spaces

`<units>` will be the units in which the ingredient is measured. Solid ingredients will be **g** (grams) or **kg** (kilograms, 1000 grams), powders or liquids in **ml** (millilitres) or **l** (litres, 1000 millilitres) while **x** will denote a simple count.

`<amount>` will be the number of the units required. The number will be positive and may contain a decimal part.

The final  $N$  lines will contain the amount of each ingredient available. The format will be the same as that described above for the ingredients, but they may not be listed in the same order. The `<amount>` value may be zero. The units will not necessarily be the same.

**Output**

If an appropriate amount of each of the ingredients required is available, the output should be

You may make your `<name>`.

where `<name>` is the name of the item from the input.

If there is not enough of one or more ingredients, the output should be

You may NOT make your `<name>`.

You need `<ingredient list>`.

where `<name>` is the name of the item from the input and

`<ingredient list>` contains the names of each ingredient for which not enough is available. Ingredients are to be listed in alphabetical order with a comma and space between each.

**Turn over for sample input and output**

### Sample Input 1

Basic butter cake  
4  
butter, g, 200  
caster sugar, g, 200  
eggs, x, 4  
self raising flour, g, 200  
eggs, x, 10  
butter, g, 500  
self raising flour, kg, 1.5  
caster sugar, g, 450

### Sample Input 2

Honey-glazed ham  
6  
ham, kg, 2  
honey, ml, 150  
orange juice, ml, 75  
soy sauce, ml, 18  
allspice, ml, 6  
ground cloves, ml, 3  
orange juice, ml, 0  
allspice, ml, 17  
ham, kg, 2  
honey, ml, 250  
ground cloves, ml, 8  
soy sauce, ml, 25

### Output for Sample Input 1

You may make your Basic butter cake.

Note that 1.5 kg of flour is 1500 g so more than enough to make the cake.

### Output for Sample Input 2

You may NOT make your Honey-glazed ham.  
You need orange juice.



**PROBLEM D**

**CROQUET**

**3 POINTS**

Association Croquet is a game played between 2 teams either as singles or doubles. Each team has 2 balls which they hit with their mallets. During a turn, the same ball must be hit each time.



The aim is to score points by manoeuvring both balls on your side through 6 hoops, once in each direction, then hitting the centre peg. A point is scored each time a ball is hit through the next hoop, and when the ball hits the centre peg. 26 points wins the game.

A number of strokes are possible; they are represented by the following codes.

- M Miss – a ball does not hit another ball, does not hit the centre peg, nor does it run a hoop. This ends the player’s turn. Play continues with a player from the opposing team.
- R A roquet. A ball hits another ball; this gives the player another hit which must be a croquet shot.
- C A croquet shot. The player’s ball is placed in contact with the ball it roqueted and played so that both balls move. The player then has a free hit.
- H A hoop shot. The ball is hit through the next hoop in the correct direction. The player scores a point and then has a free hit.
- P A ball is pegged out by hitting it against the centre peg. This scores a point and ends the player’s turn. The pegged out ball is removed from the game, and play continues with a player from the opposing team.

This problem requires you to follow a series of strokes in a game and track the score.

**Input**

The first 2 lines contain the names of the two teams who are playing the game recorded, one per line. Names may contain spaces or punctuation but will not be longer than 20 characters.

The next line contains the score at the start of the series of recorded strokes. It will consist of 2 integers, both in the range 0 to 25 inclusive, separated by a space. The first score will be for the first named team.

The fourth line will be a positive integer, N, being the number of characters in the final line ( $0 < N \leq 255$ ).

The final line will consist of a series of N characters, each being one of the letters M, R, C, H and P. These are codes for strokes as defined above. The first stroke is played

by the first named team. The sequence of strokes will not necessarily complete a game, but will not take a team's score beyond 26 points.

### **Output**

The score at the end of the sequence of strokes recorded in the format

team\_1 x team\_2 y

team\_1 and team\_2 are the names of the 2 teams in the order they were input.

x is the score of team\_1 and y is the score of team\_2

### **Sample Input**

Sam and Tom

Trish & Kathy

3 12

27

RCRCHRCHRCHRCMMRCRCHRCHRCRCH

### **Output for Sample Input**

Sam and Tom 8 Trish & Kathy 12

### **Explanation**

RCRCHRCHRCHRCM This is the first turn where 3 hoops are run by Sam and Tom.

M This is the second turn where no hoops are run by Trish & Kathy

RCRCHRCHRCRCH This is the third turn where 2 hoops are run by Sam and Tom.  
The turn is not finished but no more was recorded.

# New Zealand Programming Contest 2019

## PROBLEM E

## NETBALL COACHING

10 POINTS

Netball has had a high profile recently in New Zealand. The men's team beat the women's team (the Silver Ferns) in a recent series. The Silver Ferns then went on to win the World Championship. This has led to an upsurge in the number of people who want to take part in netball coaching classes.

Alison Cooke runs a highly rated course at various centres across New Zealand. To pass the course, a student must participate in 5 exercises for which they will be graded.

Performance in each exercise is given a pass/fail grade, and must be passed to score a mark above 0. A mark of 10 is awarded if an exercise is passed first time, 7 if passed second time and 4 if passed after 3 or more attempts.

Exercises are weighted according to how difficult Ms Cooke judges them to be. This will vary from course to course, but each exercise will be weighted between 1 and 3. The student's mark is multiplied by the weighting of the exercise to give their weighted mark for each exercise. Exercise marks are added to give each student a total weighted mark for the assessment.

To pass a student's total weighted mark must be at least 50% of the total weighted marks available for the 5 netball exercises.

### Input

The first line of input will contain 5 numbers, separated by spaces, which are the weightings for each of the 5 exercises. Each weighting will be an integer between 1 and 3 inclusive.

The next line will contain a single integer,  $S$  ( $0 < S \leq 25$ ), the number of students to assess. There then follow  $S$  lines, each defining one student. A line will contain no more than 32 characters. The line contains a 4 digit ID, the student's first name followed by the student's last name, all separated by spaces.

There then follows a series of lines representing the results for students in the exercises. Each line has the following format:

```
<student id> <exercise> <result>
```

where <student id> is the 4 digit ID of the student whose result is shown,

<exercise> is one of the letters A to E to identify the exercise,

<result> is one of the letters P or F to indicate Pass or Fail

A student who fails an exercise may take it again until it is passed.

The final line will be

```
0 # #
```

This line marks the end of input and should not be processed.

## Output

Output consists of one line for each student from the input, in the same order as they were input. Each line will contain the name of the student followed by a space, then the word Pass or the word Fail.

Sample Input	Output for Sample Input	Explanation
1 1 2 3 3 4 1206 Angela Williamson 1244 Barry Read 3564 Josephine Henry 2272 Thomas Hansen 3564 A P 1206 C F 1244 B F 3564 C F 1244 B F 1244 D F 2272 A F 1206 B F 2272 B F 1206 E P 2272 A P 1244 A P 1206 D P 3564 D P 1244 C P 2272 C F 1244 D P 3564 C F 2272 C P 3564 B P 3564 C P 1244 B F 2272 E F 2272 B P 1206 A F 1206 B P 2272 E P 0 # #	Angela Williamson Pass Barry Read Pass Josephine Henry Pass Thomas Hansen Fail	Total possible mark is $10+10+20+30+30$ ie 100 <b>Angela</b> Exercise A Failed 0 Exercise B Second time $1 \times 7 = 7$ Exercise C Failed 0 Exercise D First time $3 \times 10 = 30$ Exercise E First time $3 \times 10 = 30$ Total 67 Pass <b>Barry</b> Exercise A First time $1 \times 10 = 10$ Exercise B Failed 0 Exercise C First time $2 \times 10 = 20$ Exercise D Second time $3 \times 7 = 21$ Exercise E Not taken 0 Total 51 Pass <b>Josephine</b> Exercise A First time $1 \times 10 = 10$ Exercise B First time $1 \times 10 = 10$ Exercise C Third time $2 \times 4 = 8$ Exercise D First time $3 \times 10 = 30$ Exercise E Not taken 0 Total 58 Pass <b>Thomas</b> Exercise A Second time $1 \times 7 = 7$ Exercise B Second time $1 \times 7 = 7$ Exercise C Second time $2 \times 7 = 14$ Exercise D Not taken 0 Exercise E Second time $3 \times 7 = 21$ Total 49 Fail

New Zealand Programming Contest 2019

**PROBLEM F**




**SET GAME**

**10 POINTS**

Set is a card game played with a pack of 81 unique cards. Each card in the pack has 4 properties, having one of the three possible values for each property.

<b>Colour</b> – Red (R), Green (G), Blue (B)	<b>Number</b> 1, 2, 3,
<b>Shape</b> – Diamond (D), Oval (O), Squiggle (S)	<b>Fill</b> – Filled (F), Shaded (S), Empty (E)

This sample contains 3 cards and illustrates all the variations. If it is not in colour, note that the left card is red, the middle card is green and the right card is blue.

		
RD1F	GO2S	BS3E

Below each card is its representation, being the value for its colour, shape, number and fill in that order. In this problem, cards will always be represented in this manner.

The 3 cards illustrated form a set. A set is a collection of 3 cards such that for each property all 3 cards are either the same or different. The 3 cards above are different for all 4 properties.

Because each card is unique, at least one property must be different.

Here are some other examples.

RD1F	RD1S	RD1E	SET. Same colour, shape and number, different fill.
RD1F	RD1S	RD2E	NOT a set. Two are 1s but one is not.

It is also true that given any two cards, only 1 other card will complete a set with them. In this problem you will be given 2 cards and have to identify the 3<sup>rd</sup> card required to make a set.

**Input**

Input begins with a single integer, N, on a line by itself (0 < N <= 30). There then follow N lines, each line representing a group of 2 cards. Each card will be represented by 4 characters as described above. The cards will each be separated by a space.

**Output**

For each line of input, produce one line of output containing the 4 characters, in the specified order, that identify the other card required to form a set.

### **Sample Input**

4

RD1F RD1S

RD1F RD2E

RS1F G02F

BS3E BD1E

### **Output for Sample Input**

RD1E

RD3S

BD3F

BO2E

**PROBLEM G**

**POSTFIX OPERATIONS**

**10 POINTS**

To correctly evaluate an expression like  $3 + 4 * 2$ , you need to know the rules of arithmetic. You do the multiplication first, giving 8, then add the 3 to get 11. If you add first to get 7 then multiply that by 2 you get 14, an incorrect answer.

An alternative way to write this expression is  $3 4 2 * +$ . Here the operators come after the operands (the numbers), hence postfix. To evaluate this, when you encounter the  $*$ , multiply together the previous two operands (so  $2 * 4$ ) and store the answer. You next encounter the  $+$  so add the stored answer, 8, to the 3. This is the same as the answer in the first paragraph.

In this problem you will be required to evaluate expressions which use postfix notation.

**Input**

The first line of input will be a single integer,  $N$  ( $0 < N \leq 20$ ), the number of expressions to follow. The remaining  $N$  lines will each contain a valid expression for you to evaluate.

Each expression will contain a list of non-negative integers less than 100 and arithmetic operators ( $+ - * /$ ) with no more than 30 characters. Every operator will have two available operands.  $/$  is integer divide (returns only the integer part of the quotient). Division by zero will not occur.

**Output**

For each expression in the input, output the evaluation of the expression, a single integer.

Sample Input	Output for Sample Input	Explanation
3		
3 4 2 * +	11	$2 * 4 + 3$
2 9 5 + * 3 -	25	$(9 + 5) * 2 - 3$
17 10 + 3 * 9 /	9	$(17 + 10) * 3 / 9$





**PROBLEM H**

**ARRIVAL**

**10 POINTS**

If you have ever travelled internationally by plane, you will be aware that working out when you are going to arrive at your destination depends on the flight time and the time zone difference between departure and destination airports.

In this problem you will be given the departure city, its time zone and the departure time, the destination city and its time zone, and the flight time. From this data you must calculate the arrival time.

**Input**

Input will be for a single journey in the following format:

Line 1: The departure city

Line 2: The time of departure as mm dd hh:mm (month, day, hours and minutes all as 2 digits and separated by spaces and a colon as shown).

Line 3: The time zone of the departure city as (UTC) + or -hh:mm

Line 4: The destination city

Line 5: The time zone of the destination city as (UTC) + or -hh:mm

Line 6: The flight time as hh:mm (hours and minutes both as 2 digits and separated by a colon)

A time zone which is exactly UTC will be shown as 00:00. Flights have been selected so that:

- No flight crosses the International Date Line.
- Flights arrive on the day of departure or the following day.

**Output**

Two lines giving the city and time of departure and the city and time of arrival. The lines will start with the words *Departs* and *Arrives* respectively.

Departure time to be shown as mm dd hh:mm , as in the input.

Arrival time to be shown as hh:mm (hours and minutes, both as 2 digits and separated by a colon) followed by either *same day* or *following day* as appropriate.

**Turn over for Sample Input and Output**

### Sample Input

Dubai  
09 11 10:05  
+4:00  
Auckland  
+12:00  
15:45

### Output for Sample Input

Departs Dubai 09 11 10:05  
Arrives Auckland 09:50 following day

### Explanation

The flight took 15 hours and 45 minutes and there was an 8 hour time difference. It arrived in Auckland the following day at 9:50 am.

## PROBLEM I

## MASTERMIND

30 POINTS

*Mastermind* is a two-player board game where one player (the “codemaker”) chooses a secret *codeword* and the other player (the “codebreaker”) needs to determine the codeword by making a series of guesses. The codeword is a pattern of four coloured *code pegs*. Each peg has one of six possible colours, represented here by the letters A to F.

The codebreaker needs to determine the colours of the pegs in the codeword by guessing patterns of 4 pegs, to which the codemaker must respond with the number of *black hits* and *white hits*. One black hit is scored for each peg which has the same colour and position in both the codeword and the guessed pattern. One white hit is scored for each peg in the guessed pattern that has a corresponding peg in the codeword of the same colour but in a different position.

For example, if the codeword is FBAF and the guess is FBDA then the score is 2 black hits (from the first two pegs) and 1 white hit (from the A peg). Each codeword peg can only be counted once, so for example if the codeword is FBAF and the guess is BAAB then the score is 1 black hit (from the third peg) and 1 white hit (from one of the B pegs).

Given the guesses made by the codebreaker and the corresponding scores, your task is to determine the secret codeword and also report after how many of the guesses the codeword can be uniquely determined.

**Input**

The first line will consist of an integer  $N$ , the number of guesses ( $1 \leq N \leq 30$ ).

The following  $N$  lines will describe the guesses. Each line will consist of a guessed pattern, the number of black hits, and the number of white hits.

It is guaranteed that the codeword can be uniquely determined from the guesses and scores.

**Output**

On the first line output the codeword.

On the second line output an integer indicating after how many of the guesses the codeword could be uniquely determined.

*Turn over for sample input and output.*

### Sample Input 1

```
5
FBDA 2 1
BAAB 1 1
BCDA 0 2
FBAE 3 0
FBAF 4 0
```

### Output for Sample Input 1

```
FBAF
4
```

### Explanation

The response to the first guess shows that two of the letters are correct and in the correct place, and that there is one correct letter in the wrong place. The fully correct two are the first two, and the partially correct one is the fourth (although the codebreaker does not have enough information yet to know this). Continuing with the guesses and responses the codebreaker learns more about the codeword until they can deduce that it is FBAF after four guesses (the last guess wasn't really required).

### Sample Input 2

```
11
AAAA 1 0
BBBB 0 0
CCCC 0 0
DDDD 0 0
EEEE 1 0
AEFF 2 2
FFAE 1 3
AFFE 2 2
FEAF 1 3
FAFE 0 4
EFAF 2 2
```

### Output for Sample Input 2

```
AFEF
8
```

## PROBLEM J

## GOING FISHING

30 POINTS

There are  $N$  fish in the ocean. The  $i$ -th fish starts at integer coordinates  $(x_i, y_i)$  with  $y_i \geq 1$  and swims parallel to the X-axis (either in the positive X direction or negative X direction) at a fixed speed of 1 unit per second.

A fishing boat needs to collect as many fish as possible. It can start anywhere along the X-axis; that is, its initial  $y$  coordinate must be 0 and its initial  $x$  coordinate can be any real number (doesn't have to be an integer). The boat can only sail parallel to the Y-axis in the positive Y direction. The boat can wait before starting, but once it starts it must sail at a fixed speed of 1 unit per second without stopping.

The boat is considered to collect a fish if the boat and the fish are at the same location at some instant of time. What is the maximum number of fish that the boat can collect?

**Input**

The first line will contain a single integer  $N$ , the number of fish ( $1 \leq N \leq 100,000$ ).

Each of the following  $N$  lines will describe a fish. The  $i$ -th of these lines will contain three space-separated integers  $x_i$ ,  $y_i$  and  $v_i$  giving the location of the  $i$ -th fish ( $-10^9 \leq x_i \leq 10^9$ ,  $1 \leq y_i \leq 10^9$ ) and its direction:  $v_i = 1$  means the fish swims in the positive X direction and  $v_i = -1$  means the fish swims in the negative X direction.

**Output**

Output a single integer, the maximum number of fish that the boat can collect.

**Sample Input**

```
3
-3 1 1
3 2 -1
8 3 -1
```

**Output for Sample Input**

```
2
```

**Explanation**

The boat can catch the first two fish by starting from  $x = -0.5$  after waiting 1.5 seconds.



## PROBLEM K

## SEQUENCE PERIODS

30 POINTS

John hates maths class. It seems like all his teacher does is write numbers on the blackboard. For example, yesterday she wrote the sequence

71 4 24 71 48 71 4 24 71 48 71 4 24 71 48 71 4 24

In his boredom, John noticed that those are the same five numbers repeated over and over again. The number five is said to be the *least period* of the sequence.

Your task is to help John work out the least period of some sequences.

Formally, for a sequence of length  $N$  with elements  $A[0], \dots, A[n-1]$ , the least period is the smallest positive integer  $P$  such that  $A[i+P] = A[i]$  for all  $i$  where  $0 \leq i < N-P$ .

**Input**

The first line will contain a single integer  $N$ , the length of the sequence ( $1 \leq N \leq 1,000,000$ ).

The second line will contain  $N$  space-separated integers  $A[0], \dots, A[n-1]$ , the elements of the sequence ( $0 \leq A[i] \leq 1,000,000$  for all  $i$  where  $0 \leq i < N$ ).

**Output**

Output a single integer, the least period of the sequence.

**Sample Input 1**

18  
71 4 24 71 48 71 4 24 71 48 71 4 24 71 48 71 4 24

**Output for Sample Input 1**

5

**Sample Input 2**

6  
42 100 99 42 42 100

**Output for Sample Input 2**

4

**Sample Input 3**

6  
1 2 3 1 3 2

**Output for Sample Input 3**

6

**Explanation**

For the second sample, the least period is considered to be 4 even though the sequence ends before the first full repetition.

For the third sample, the sequence is not considered to repeat, but technically  $N$  is always a period.





**PROBLEM L****PASSWORD REQUIREMENTS****30 POINTS**

Websites often place requirements on passwords in an effort to force users to pick strong passwords. For example, a website might require passwords to contain at least 2 digits, at least 1 upper case letter, and at least 1 lower case letter.

We would like to generate a password of length  $L$  by finding the  $K$ -th valid password in lexicographic (dictionary) order. The allowed characters are, in lexicographical order,  $0-9$ ,  $A-Z$ , and  $a-z$ .

**Input**

The first line will contain a single integer  $L$ , the desired password length ( $1 \leq L \leq 10$ ).

The second line will contain three space-separated integers  $D$ ,  $U$ , and  $S$ : the password requirements. A password is valid if it contains at least  $D$  digits, at least  $U$  upper case letters, and at least  $S$  lower case letters. It is guaranteed that  $0 \leq D, U, S \leq L$  and  $D + U + S \leq L$ .

The third line will contain a single integer  $K$  satisfying  $1 \leq K \leq N$ , where  $N$  is the number of valid passwords. It can be shown that  $N < 2^{60}$  when  $L \leq 10$ .

**Output**

Output the  $K$ -th valid password of length  $L$ .

**Sample Input**

```
4
2 1 1
3
```

**Output for Sample Input**

```
00Ac
```

**Explanation**

The desired password length is 4. The requirements match the example from the introduction. There are 811,200 valid passwords, the first is 00Aa and the last is zZ99. The sample input asks for the third, which is 00Ac.



## PROBLEM M

## SURFING

100 POINTS

The New Zealand Paihia Surfing Club (NZPSC) encourages members to surf for fun. Indeed, it expects members to optimise their surfing to maximise fun.

The club measures time from the start of the day in 'wave period's ( $wp$  for short). At  $time = 1$ , the first wave arrives at the beach. At  $time = 2$  the second wave arrives, and so forth. Distance is measured in 'wave length's ( $wl$  for short), being the distance between wave peaks. The conditions at the club's beach are such that  $wp$  and  $wl$  are constants: the waves arrive at a steady rate, and are a steady distance apart before reaching the beach. Waves travel toward the beach at a speed of  $1 wl/wp$ .

Sophisticated software which the club's experts run each day provides in advance a list of the wave heights to be expected ( $H_t$ ) for waves arriving at times from  $1 wp$  to  $N wp$ . (Special note:  $time = 0$  is after 1pm in the afternoon, to allow for the day/night cycles of the club's members who are all software developers.)

The club requires adherence to a surfing style guide. Surfers do a sequence of trips: each trip must start at an integer time ( $\geq 0$ ). The surfer swims out at a speed of  $1 wl/wp$  for an integer number of wave periods and meets a wave at an integer distance (in  $wl$ ) from the beach. They then ride the wave all the way back to the beach. Safety requirements impose a strict upper limit of  $K wl$  on the distance from the shore at which a wave is caught. Members may start swimming out again immediately on return to the beach, or they may wait for one or more  $wp$  between rides. To avoid exhaustion, an upper limit of  $D wl$  is set on the total distance swum by a member during a day's surfing.

A fun level is calculated for each ride - being the length of the ride (in  $wp$ ) multiplied by the height of the wave being ridden. Members are required to choose waves in order to maximise the sum of fun levels for their rides. Your problem is to determine the maximum fun that can be obtained given values of  $N$ ,  $D$ ,  $K$  and  $H_t$  ( $1 \leq t \leq N$ ).

**Input**

The first line will contain three space-separated integers:  $N$ ,  $D$  and  $K$  where  $1 \leq N \leq 1000$ ,  $1 \leq D \leq 100$ , and  $1 \leq K \leq 10$ .

Each of the following  $N$  lines contain a single integer being the heights of waves 1, 2, 3, ...  $N$  respectively. Wave heights will be in the range 0 to 1000 (inclusive).

**Output**

Output a single integer, the maximum fun that can be achieved given the input values.

*Turn over for sample input and output.*

### Sample Input

14 5 3  
5  
5  
5  
5  
5  
9  
5  
5  
5  
11  
5  
5  
5  
10

### Output for Sample Input

53

### Explanation

The most fun is achieved by waiting on the beach until time 4, swimming 3 to meet wave 10, surfing back 3 for fun 33; swimming 2 to meet wave 14, surfing back for 2 for fun 20; giving a total distance swum of 5 and total fun of 53.

**PROBLEM N****PARCELS****100 POINTS**

There are  $N$  people numbered  $0$  to  $N-1$  trying to play a game of "pass the parcel". Each person has a fixed *target person* to which they pass parcels: when person  $i$  receives a parcel they pass it to person  $T_i$ . Unfortunately, they haven't managed to form a neat circle and it's all a bit of a mess.

Given the configuration of people, we wish to make a series of  $Q$  queries. For the  $j$ -th query, we would like to know whether a parcel given to person  $A_j$  will ever reach person  $B_j$ , and if so, how many passes it will take.

**Input**

The input will contain a description of the people's configuration and multiple queries about that configuration.

The first line will consist of two space-separated integers  $N$  and  $Q$ , the number of people and the number of queries ( $2 \leq N \leq 100,000$ ,  $1 \leq Q \leq 100,000$ ).

The second line will consist of  $N$  space-separated integers  $T_0, \dots, T_{N-1}$ , the targets of the people ( $0 \leq T_i \leq N-1$ ,  $T_i \neq i$ ).

Each of the following  $Q$  lines will describe a query. The  $j$ -th of these lines will consist of two space-separated integers  $A_j$  and  $B_j$  ( $0 \leq A_j \leq N-1$ ,  $0 \leq B_j \leq N-1$ ,  $A_j \neq B_j$ ).

**Output**

Output one line for each query. On the  $j$ -th line, output the number of times a parcel given to person  $A_j$  will be passed until it reaches person  $B_j$ , or  $-1$  if the parcel will never reach person  $B_j$ .

**Sample Input**

```
5 2
1 0 1 4 3
2 0
3 1
```

**Output for Sample Input**

```
2
-1
```

**Explanation**

For the first query, if person 2 is given a parcel they will pass it to person 1 who will then pass it to person 0 (taking two passes).

For the second query, if person 3 is given a parcel they will pass it to person 4 who will then pass it back to person 3. This will repeat forever and the parcel will never reach person 1.



**PROBLEM O****BLINKERS****100 POINTS**

Have you ever waited at a traffic light and noticed the indicators of the car in front of you blink in sync with your car's indicators, then slowly go out of sync? If you were to wait long enough they'll eventually blink in sync again.

Suppose the blinkers of one car have an integer frequency of  $A$  blinks per minute and the blinkers of another car have an integer frequency of  $B$  blinks per minute. We can count the number of times in the first minute that the blinkers begin their flash at exactly the same moment, assuming their first flash began at the start of the minute.

For example, if  $A$  is 8 and  $B$  is 6, they will blink together once at the start and once at the 30 second mark. The number of times they blink together in the first minute is 2. They'll also blink together at the 60 second mark, but that isn't considered to be in the first minute.

Randall thinks the formula for the number of times they blink together in the first minute is  $A - B$  (assuming  $A > B$ ), but the formula doesn't always work. For example, if  $A$  is 8 and  $B$  is 5 they will only blink together once at the start, but  $A - B$  is 3.

Given a fixed value of  $A$ , for how many of the frequencies  $B$  in the range 1 to  $A-1$  does Randall's formula give the correct answer?

**Input**

The input will be a single line with the integer  $A$  ( $2 \leq A \leq 10^{12}$ ).

**Output**

Output a single integer: the number of values of  $B$  in the range 1 to  $A-1$  for which two blinkers with frequencies  $A$  and  $B$  will blink together exactly  $A - B$  times in the first minute.

**Sample Input**

8

**Output for Sample Input**

3

**Explanation**

The values of  $B$  for which Randall's formula gives the correct answer are 4, 6, and 7.





## PROBLEM P

## XOR CIPHER

100 POINTS

I have a message that was encrypted using a simple XOR cipher. I don't know the key or the original message, but I do know that each of the characters in the original message was in a particular range. How many possible keys are there?

The message consists of  $N$  characters. A character is a  $W$ -bit integer (that is, an integer in the range  $0$  to  $2^W - 1$ ). The key is also a  $W$ -bit integer.

The message was encrypted by XOR-ing each character with the key. Recall that to XOR two numbers, they are written in binary, then the  $i$ -th bit of the result is 1 if the  $i$ -th bits of the two numbers are different and 0 if they are the same. For example,  $12 \text{ XOR } 5 = 9$  (in binary:  $1100 \text{ XOR } 0101 = 1001$ ).

The characters in the original message are known to be in the range  $A$  to  $B$  (inclusive). The characters in the encrypted message are  $C_1 .. C_N$ .

**Constraints**

$$1 \leq N \leq 1,000$$

$$1 \leq W \leq 62$$

$$0 \leq A \leq B \leq 2^W - 1$$

$$0 \leq C_i \leq 2^W - 1$$

**Input**

The first line will consist of two space-separated integers  $N$  and  $W$ .

The following line will consist of two space-separated integers  $A$  and  $B$  (in decimal).

$N$  lines follow. The  $i$ -th of these lines contains  $C_i$  (in decimal).

**Output**

Output the number of possible keys. There might not be any possible keys, in which case output 0.

**Sample Input**

```
2 3
3 5
5
2
```

**Output for Sample Input**

```
2
```

**Explanation**

In binary,  $A$  is  $011$ ,  $B$  is  $101$ , and encrypted message is  $101\ 010$ .

The possible keys in binary are  $001$  (corresponding to an original message of  $100\ 011$ ) and  $110$  (corresponding to an original message of  $011\ 100$ ).

