South African Computer Olympiad
Training Camp 3, 2005
Day 1

Overview

Author	**Timothy Stranex**	**Migael Strydom**	**Keegan Carruthers-Smith**	**Nick Pilkington**
Problem	**morse**	**gear**	**war**	**phone**
Source	morse.c morse.cpp morse.pas	gear.c gear.cpp gear.pas	war.c war.cpp war.pas	phone.c phone.cpp phone.pas
Input file	morse.in	gear.in	war.in	phone.in
Output file	morse.out	gear.out	war.out	phone.out
Time limit	2 seconds	1 second	1 second	1 second
Number of tests	10	10	10	10
Points per test	10	10	10	10
Total points	**100**	**10**	**100**	**100**

The maximum total score is 310 points.

South African Computer Olympiad
Training Camp 3, 2005
Day 1

Morse Codes

Author

Timothy Stranex

Introduction

Auron the Adventurist was out adventuring in a seaside cavern when he accidentally dislodged a rock which caused the entrance to be blocked. To make matters worse, sea water has begun to leak into the cavern so that in only a few minutes, he will be fully submerged. Clearly, he will die. As such, he wants to make sure that his death in his selfless persuit of knowledge is recorded in Wikipedia. Luckily, Auron carries arround a modified Morse transmitter for this express purpose.

Task

The transmitter can transmit two different types of pulses, dot or a dash. A sequence of these pulses is called a *code*. Auron wants to assign a code to each of the letters $l_1, l_2, \ldots, l_L$ in his alphabet. Furthermore, codes must be *prefix-free*; that is, no code may be a prefix of any other. For example, if $L = 2$, then

Letter	Code
l_1	$dot \cdot dash$
l_2	$dash \cdot dot$

is valid, but

Letter	Code
l_1	$dot \cdot dash$
l_2	$dot \cdot dash \cdot dot$

is invalid because $dot \cdot dash$ is a prefix of $dot \cdot dash \cdot dot$.

Auron has modified his transmitter so that the length of a dot is d_1 and the length of a dash is d_2. Auron wants to assign a code to each letter in the most efficient possible way. He has not yet decided what message to send, so choose an assignment that minimises the time required to send each letter once.

Example

Suppose $L = 3$ and the pulse lengths are $d_1 = 1$ and $d_2 = 3$. Then the best choice of codes is

Letter	Code
l_1	$dash$
l_2	$dot \cdot dash$
l_3	$dot \cdot dot$

For this choice, transmitting the message $l_1 \cdot l_2 \cdot l_3$ takes nine units of time. In this case, no other choice is faster.

Input (morse.in)

The first line of the input contains three space-separated integers, L, d_1 and d_2.

Sample input

```
3 1 3
```

Output (morse.out)

The output contains L lines. Each line contains several space-separated integers. The first integer on line i is C_i, the number of pulses in the code for letter l_i. The next C_i integers on that line are either 1 which means a dot pulse or 2 which is a dash pulse. There may be several solutions to the problem, you only need to output one of them.

Sample output

```
2 1 1
2 1 2
1 2
```

Constraints

- $2 \leq L \leq 100000 = 10^5$
- $1 \leq d_1, d_2 \leq 1000$

50% constraints

- $2 \leq L \leq 20$

Time limit

2 seconds.

Scoring

If the solution in the output is optimal, the program will score 100% for the test-case. If the codes are not prefix-free, the solution is not optimal or the output is not formatted correctly, the program will score 0% for the test-case.

South African Computer Olympiad
Training Camp 3, 2005
Day 1

Gear Counting

Author

Migael Strydom

Introduction

You are trying to construct Charles Babbage's Analytical Engine, but have run into a slight problem with the gears of the machine. The gears are all connected in a long row, but each gear has a different number of teeth. The gears are numbered from 1 starting at the left to N at the right, where N is the number of gears. Each gear has one tooth that is coloured red. During the operation of the machine, the first gear will rotate, which in turn rotates all the other gears at one tooth per second. Initially, that is, zero seconds after the machine has started, each gear's red tooth will point to the top. Thereafter, for each individual gear i, where $1 \leq i \leq N$, the red tooth will point to the top every T_i seconds, where T_i is the number of teeth of gear i.

Task

Your task is to count the number of times in a given time interval one or more of the gears have their red tooth pointing to the top.

Example

Figure 1 shows an example with three gears in their starting position at time $t = 0$. Suppose you need to count the number of times there is a red gear pointing to the top in the time interval between $t = 15$ and $t = 25$ seconds, including 15 and 25. For gear 1, this will happen at $t = 16$ and $t = 24$. For gear 2 it happens at $t = 15$, $t = 20$ and $t = 25$. For gear 3 it happens at $t = 18$ and $t = 24$. The total number of times there is at least one red tooth pointing to the top is therefore 6: twice for gear 1, three times for gear 2 and two more times for gear 3. However, at $t = 24$ both gears 1 and 3 have a red tooth pointing to the top, and they are only counted once.

Input (gear.in)

The first line contains a single integer N, the number of gears. The next line contains two integers, t_1 and t_2, the time interval between which the number of times a red tooth points to the top needs to be counted, including both end points. The next N lines each contain a single integer T_i, the number of teeth that gear i has.

Sample input

```
3
15 25
8
5
6
```

Output (gear.out)

You should output a single line containing the integer that is the answer to the task.

Sample output

```
6
```

Constraints

- $1 \leq N \leq 15$
- $0 \leq t_1 \leq t_2 \leq 10^{15}$
- $3 \leq T_i \leq 100$

50% constraints

- $0 \leq t_1 \leq t_2 \leq 10000$

Time limit

1 second.

Scoring

You will score 10 points for a correct solution and 0 points for an incorrect solution.

South African Computer Olympiad
Training Camp 3, 2005
Day 1

War

Author

Keegan Carruthers-Smith

Introduction

After tinkering with your time machine, you accidentally turn it on. You are now in the dark ages and have been captured by one of the kingdoms. While bargaining for your life, you mention that you have the ability to solve problems for them. This is great for them because they are planning an attack on another kingdom and are having problems determining how many soldiers they can send.

Task

You have to help plan the attack to save your life. You are given details of the map of the local area. The map shows paths connecting different villages. Some paths are more dangerous than others so you are also given the amount of soldiers allowed to use each path. You must give the largest possible amount of soldiers that can be sent in one go for the attack.

One of the war advisors gives you some extra information that he thought would be helpful. He told you that no village (including the starting location) has more than 5 paths leading from it, and that the soldiers can move in either direction along the path. He also said that for any pair of villages, there is at most one path between them.

Example

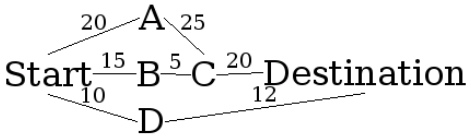

Figure 1: Sample Map

Refer to figure 1 for a sample map. If you send:

- 15 soldiers along the path Start → A → C → Destination
- 5 soldiers along the path Start → B → C → Destination
- 10 Soldiers along the path Start → D → Destination

You will have the maximum amount of soldiers (30) sent for the map. There are other configurations for this maximum amount, but we are only concerned with the maximum amount possible.

Input (war.in)

The first line contains the two integers N and M, the number of villages (including the start and destination) and the number of paths respectively. The next M lines each contain three integers, a_m b_m c_m. a_m and b_m represent a path between two villages and c_m represents the amount of soldiers allowed to go along the path.

The start point is always 1 and the destination point is always N.

Sample input

```
6 7
1 2 20
1 3 15
1 4 10
2 5 25
3 5 5
4 6 12
5 6 20
```

Output (war.out)

The output contains a single line containing the maximum number of soldiers that can be sent.

Sample output

```
30
```

Constraints

- $2 \leq N \leq 900$
- $1 \leq M \leq 2200$
- $1 \leq a_i, b_i \leq N$
- $1 \leq c_i \leq 10000$

50% constraints

- $2 \leq N \leq 100$
- $1 \leq M \leq 250$

South African Computer Olympiad
Training Camp 3, 2005
Day 1

Time limit

1 second.

Scoring

You will score 100% for a correct answer and 0% for an incorrect answer.

South African Computer Olympiad
Training Camp 3, 2005
Day 1

Telephone

Author

Nick Pilkington

Introduction

A long time ago in a galaxy far, far away, an Alien race was introduced to cellphones. Suddenly interplanetary smsing became hugely popular! The Alien race is such that any excess physical exercise causes them to explode; a most unfortunate consequence of being on a planet with no atmosphere. Each key of the Alien phones is assigned a certain number of letters of their alphabet. But because there are more letters in their alphabet than keys on the phone, extra presses are needed to generate some letters. Consider the keypad shown in Table 1, which has been translated into EarthSpeak:

1 ABC	2 DEF	3 GHI
4 JKL	5 MNO	6 PQR
7 STU	8 VW	9 XYZ

Table 1: An example Alien keypad

To generate a 'D' you'd simply press 2 once, however an 'E' would require two presses and an 'F' three and so forth...

The Alien race, fearing for their own survival, have recalled all phones on the planet and would like to consider the letter to key assignment so as to put the users through as little physical strain as possible. You have been outsourced to solve the problem of assigning the letters to keys of their phone.

Task

You will be told the number of keys on the Alien phone and the number of letters in the Alien alphabet. You will then be given the frequency that each of the letters appears in the language. You will have to calculate which letters are assigned to which keys such that the value of the keypad is minimized and adheres to the following constraints:

If read from left to right, top to bottom on the key pad the letters should appear in order. The value of the keypad is given by the sum of each letters frequency multiplied by its position on the key from the left.

The keys and letter will be given as numbers for simplicity!

Example

Consider a keypad with two buttons and 3 letters to assign to it. The first letter has frequency 2, the second has frequency 8, and the third has frequency 1. In this case, it would be best to assign the first letter to the first key, and the remaining two letters to the second key for a total value of $(1 \cdot 2) + (1 \cdot 8) + (2 \cdot 1) = 12$.

Input (phone.in)

- **Line 1:** Two positive integers K, L. K represents the number of keys on the phone $1..K$. L represents the number of letters in the alphabet $1..L$.
- **Line 2..L+1:** Each contains one integer F that represents the frequency of the corresponding letter corresponding to that line.

Sample input

```
2 3
2
8
1
```

Output (phone.out)

The output should contain K lines of output corresponding to the K keys on the phone. Each line should contain some number of space separated integer in increasing order that represent the letters assigned to this key.

Sample output

```
1
2 3
```

Constraints

- $1 \leq K < L \leq 1000$
- $1 \leq F \leq 1000$ for any frequency F

50% constraints

- $1 \leq K < L \leq 30$
- $1 \leq F \leq 1000$ for any frequency F

Time limit

1 second.

Scoring

If the output is in any way invalid you will score 0 and be solely responsible for the spontaneous combustion of countless innocent Aliens. If the output reflects the optimal keypad you will score 100% and be offered a job on the planet without an atmosphere, cool!