



# The 6th FanRuan Cup Southeast University Programming Contest (Winter, Advanced Group)

## Problems





## A. Line segments

time limit per test: 1 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

Little J has a planar Cartesian coordinate system with a line segment in the plane. Little J wants to find a line segment in the plane that has the same length as this segment and is perpendicular to each other, so please help him find such a line segment.

Two line segments in the same plane are said to be perpendicular to each other when and only when both segments are extended as straight lines.

### Input

In the first line, an integer  $t(1 \leq t \leq 10^4)$ , representing the number of tests.

For each test, input only one line of four integers  $x_1, y_1, x_2, y_2(-10^8 \leq x_1, y_1, x_2, y_2 \leq 10^8)$ ,  $(x_1, y_1)$  and  $(x_2, y_2)$ , representing the two endpoints of an existing line segment in the plane respectively, making sure that the two endpoints of the line segment are different.

### Output

For each test, output four integers  $u_1, v_1, u_2, v_2(-10^9 \leq u_1, v_1, u_2, v_2 \leq 10^9)$  with  $(u_1, v_1)$  and  $(u_2, v_2)$  representing the two endpoints of the line segment that you gave.

Any answer that satisfies the requirements of the question is correct.

| Sample Input 1      | Sample Output 1    |
|---------------------|--------------------|
| 4                   | -1 1 0 0           |
| 0 0 1 1             | 0 0 2 -8           |
| -3 2 5 4            | 7 -16 -7 17        |
| 17 9 -16 -5         | -11111 0 8520 -230 |
| -111 -451 119 19180 |                    |



## B. Good Array

time limit per test: 1 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

For an array  $a$  of length  $n$  and a positive integer  $m$ , the array  $a$  is said to be **good** when there exists a positive integer  $b$  such that of all positive integers less than or equal to  $m$ , **if and only if** those occurring in the array  $a$  are factors of  $b$ .

For example, when  $a=[1,2,3], m=5$  is a good array because  $b=6$  can be chosen. But when  $a=[2,3], m=5$  it **isn't** a good array because a factor 1 of  $b$  is missing from the array  $a$  when  $b=6$ . When  $a=[1,2,3,4], m=6$  it **cannot** be chosen when  $b=4$  because the number 3 in the array  $a$  is not a factor of  $b$ .

Given an array  $a$  of length  $n$  and a positive integer  $m$ , determine whether each prefix of  $a$  is a **good** array.

### Input

In the first line, an integer  $t$  ( $1 \leq t \leq 2 \times 10^4$ ), representing the number of tests.

For each test:

In the first line, two integers  $n, m$  ( $1 \leq n \leq 10^5; 1 \leq m \leq 10^7$ ), representing the length of the array  $a$ , and the meaning of  $m$  is the same as in the title.

In the second line, an array  $a_1, \dots, a_n$  ( $1 \leq a_i \leq m$ ) of length  $n$ , representing the array  $a$ .

It is guaranteed that the sum of  $n$  within the same testcase does not exceed  $10^5$ .

### Output

For each test, output a binary string  $s$  of length  $n$ . If  $s_i = 1$  means that the array  $a$  length  $i$  prefix is a good array; if  $s_i = 0$  means that the array  $a$  length  $i$  prefix is not a good array.

| Sample Input 1 | Sample Output 1 |
|----------------|-----------------|
| 4              | 011001          |
| 6 16           | 11011000        |
| 2 1 4 6 3 12   | 001             |



|                           |     |
|---------------------------|-----|
| 8 10000000                | 000 |
| 1 2 8 4 16 9999995 1024 1 |     |
| 3 5                       |     |
| 3 2 1                     |     |
| 3 6                       |     |
| 3 2 1                     |     |



## C. Mountains

time limit per test: 3 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

*I have stepped on top of mountains and into low valleys, and both have benefited me.*

$n$  Mountains line up from high to low, and that's the way into the trough; however, this is not the time to enter the trough, **you can change the height of the mountain!!!**

In each operation:

- You can select any mountain with subscript  $i$  and make its height becomes  $k$  times the current and height. Formally, you can choose any mountain  $i$  and make  $a_i$  to  $a_i = a_i \times k$ .

It is possible to be in a trough at any time, in any place. So ask, given  $k$ , what is the minimum number of operations that make the heights of all mountains in interval  $l$  to  $r$  non-decreasing.

Formally, given a monotonically non-increasing sequence  $a$  of length  $n$ , in each operation you can choose a position  $i (1 \leq i \leq n)$  and make  $a_i = a_i \times k$ . Then what is the minimum number of operations that make a certain subsegment  $[a_l, \dots, a_r]$  of the sequence  $a$  nondecreasing, given  $k$ .

### Input

In the first line, an integer  $t (1 \leq t \leq 10^4)$ , representing the number of tests.

For each test:

In the first row, three integers  $n, q, b (2 \leq n \leq 5 \cdot 10^5; 1 \leq q \leq 5 \cdot 10^5; 1 \leq b \leq 10^6)$ , representing the number of straight lines up the hill, the number of times it was asked, and the base when indicating the height of the hill.

The next line has  $n$  integers  $x_1, \dots, x_n (0 \leq x_i \leq n)$ , representing the heights of the  $i$ th mountain at the beginning  $a_i = b^{x_i}$ , which ensures that the array of heights of all mountains  $a$  is **non-increasing**.

The next consecutive  $q$  rows of three integers  $l, r, k (1 \leq l < r \leq n; 2 \leq k \leq 10^9)$  each represent the range of intervals for each query and  $k$  at the time of the query.

It is guaranteed that neither the sum of  $n$  nor the sum of  $q$  within the same testcase exceeds  $5 \cdot 10^5$ .



## Output

For each query, output a line of integers representing the minimum number of operations to make the array of heights of the mountains in the current interval monotonically undecreasing.

Because this question may have some precision issues, a relative error of  $10^{-5}$  between the player's answer and the standard answer is allowed, i.e., if the standard answer is *ans* and the player's output is *out*, then the player's output will be considered correct if and only if

$$\frac{|ans-out|}{ans} \leq 10^{-5}.$$

| Sample Input 1        | Sample Output 1 |
|-----------------------|-----------------|
| 4                     | 0               |
| 4 2 1                 | 0               |
| 1 3 0 4               | 0               |
| 1 4 2                 | 61              |
| 2 3 7 1               | 0               |
| 2 1 2                 | 12              |
| 1 1                   | 6               |
| 1 2 2 7 3             | 28              |
| 10 6 2                | 1               |
| 10 10 8 6 2 1 1 1 0 0 | 357             |
| 1 10 2                | 21              |
| 1 2 9 9 8 2 4 4 3 5 3 | 21              |
| 3 7 4                 | 20              |
| 2 5 3 1               |                 |
| 2 10 3 3              |                 |
| 2 3 1 2               |                 |
| 7 4 9 9 7 8 7         |                 |
| 6 5 4 3 2 1 0         |                 |
| 1 7 2                 |                 |
| 1 7 9 9 7 8 8         |                 |
| 1 7 9 9 7 8 7         |                 |
| 2 6 9 9 7 8 6         |                 |



## D. Protection

time limit per test: 2 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

The continent of Tivat is a tree, with each node of the tree numbered from 1, and in each node of the continent there is endless knowledge and treasure. Now, prophets say that in the near future, a viral crisis will erupt at node 1, and the virus will keep spreading to neighboring nodes along the edges of the tree. Suppose that each point has a value  $v$  and can be made uninfected at a cost of  $w$  (including node 1). Then ask what is the minimum total value of infected nodes using the optimal policy, if the sum of the expenditures available to make the node uninfected is at most  $c$ ?

### Input

In the first line, an integer  $t$  ( $1 \leq t \leq 10^4$ ), representing the number of tests.

For each test:

First line, two integers  $n, c$  ( $2 \leq n \leq 5000; 1 \leq c \leq 10^9$ ), representing the number of nodes on the continent of Tivat and the maximum value spent, respectively.

The second row,  $n$  integers  $v_1, \dots, v_n$  ( $0 \leq v_i \leq 5000; \sum v \leq 5000$ ), represents the value of each point.

The third line,  $n$  integers  $w_1, \dots, w_n$  ( $1 \leq w_i \leq 10^9$ ), represents the cost of making each point uninfected.

The next consecutive  $n - 1$  rows, each with two integers  $x, y$  ( $1 \leq x, y \leq n$ ), represent an edge in the tree, ensuring that all edges form a tree.

It is guaranteed that neither  $n^2$  nor  $(\sum v)^2$  will sum to more than  $5 \times 10^7$  within the same testcase.

### Output

For each test, output an integer representing what is the minimum total value of infected nodes with spending no more than  $c$ .



| Sample Input 1                  | Sample Output 1 |
|---------------------------------|-----------------|
| 4                               | 1               |
| 3 10                            | 0               |
| 1 1 1                           | 315             |
| 11 4 5                          | 2               |
| 1 2                             |                 |
| 1 3                             |                 |
| 4 9                             |                 |
| 1 2 3 4                         |                 |
| 9 10 11 12                      |                 |
| 1 2                             |                 |
| 2 3                             |                 |
| 3 4                             |                 |
| 11 20                           |                 |
| 5 27 40 39 47 67 69 75 77 80 30 |                 |
| 27 25 24 21 20 19 18 15 5 7 1   |                 |
| 1 2                             |                 |
| 3 2                             |                 |
| 4 2                             |                 |
| 1 5                             |                 |
| 3 7                             |                 |
| 6 7                             |                 |
| 7 8                             |                 |
| 5 9                             |                 |
| 5 10                            |                 |
| 11 5                            |                 |
| 2 1                             |                 |
| 1 1                             |                 |
| 2 2                             |                 |
| 1 2                             |                 |



## E. Time Traveler

time limit per test: 2 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

*The Time Traveler is destined to be alone.*

Doctor who is a time traveler who often travels through the universe. There are many planets in the universe, and the planets are the destinations of his travels. The universe can be viewed as a directed graph, and for two planets numbered  $u, v$  there is a directed edge from  $u$  to  $v (u < v)$  between them, and it takes  $w$  time to pass through this edge (the length of the edge is  $w$ ). At the same time, the universe is constantly changing, and for an edge of length  $w$ , this edge exists only from  $t$  to  $t + w$  time in the time axis, so Doctor who must start from the beginning of this edge at exactly  $t$  time.

Doctor who has a time machine that can travel through time, which can only be used on a planet, and each time it travels through time it consumes a certain amount of energy, if there are two points in time  $t_1, t_2$ , and the amount of energy it takes for the machine to travel from time  $t_1$  to time  $t_2$  is  $|t_1 - t_2| + c$  ( $c$  is a constant, which is different for each planet), and Doctor who's position does not change after using the time machine.

Then ask, if Doctor who can do one of the following three things at any given moment:

1. Stay on a planet.
2. move on the edge of a line connecting two planets.
3. travel through time

If he departs from planet 1 at moment 0, what is the minimum value of the sum of the moments at which he arrives at each planet and the energy consumed by the time machine.

### Input

In the first line, an integer  $t (1 \leq t \leq 10^4)$ , representing the number of tests.

For each test:

First line, two integers  $n, m (2 \leq n \leq 2 \cdot 10^5; 1 \leq m \leq 4 \cdot 10^5)$ , representing the number of planets in the universe and the number of edges connecting the planets.

The second line,  $n$  integers  $c_1, \dots, c_n (0 \leq c_i \leq 10^9)$ , represents the constant for the amount of energy consumed to time travel on each planet.



The next consecutive  $m$  rows of four integers each  $u_i, v_i, w_i, t_i (1 \leq u_i < v_i \leq n; 1 \leq w \leq 10^9; 0 \leq t_i \leq 2 \cdot 10^9)$  represent the two planets connecting the oriented edges, the length of the edges, and the time of the edges' existence at the beginning of the timeline.

Ensure that the sum of  $n$  within the same test point does not exceed  $2 \cdot 10^5$  and the sum of  $m$  within the same testcase does not exceed  $4 \cdot 10^5$ .

## Output

For each test:

Output one line with  $n$  integers, for each planet numbered from 1 to  $n$ , output an integer representing the minimum value of the sum of the moment of arrival at the planet plus the amount of energy consumed by the time machine, which is  $-1$  if Doctor who can never arrive at the planet.

| Sample Input 1                   | Sample Output 1                    |
|----------------------------------|------------------------------------|
| 4                                | 0 12                               |
| 2 1                              | 0 1 4                              |
| 9 16                             | 0 -1 9 13 -1 50 22 15              |
| 1 2 4 8                          | 0 3000000000 5000000000 7000000000 |
| 3 3                              |                                    |
| 0 1 0                            |                                    |
| 1 2 1 0                          |                                    |
| 2 3 2 0                          |                                    |
| 1 3 1 8                          |                                    |
| 8 10                             |                                    |
| 3 0 1 2 5 0 8 0                  |                                    |
| 1 3 9 0                          |                                    |
| 3 8 5 2                          |                                    |
| 7 8 11 90                        |                                    |
| 2 5 1 0                          |                                    |
| 1 8 7 15                         |                                    |
| 2 4 1 0                          |                                    |
| 3 4 3 10                         |                                    |
| 2 5 9 16                         |                                    |
| 4 6 25 25                        |                                    |
| 1 7 6 16                         |                                    |
| 4 3                              |                                    |
| 1000000000 1000000000 1000000000 |                                    |
| 1000000000                       |                                    |
| 1 2 1000000000 2000000000        |                                    |



|                  |  |
|------------------|--|
| 2 3 1000000000 0 |  |
| 3 4 1000000000 0 |  |



## F. Trick

time limit per test: 1 seconds

memory limit per test: 1024 megabytes

input: standard input

output: standard output

Ambition is out buying drinks! nkxjlym, who specializes in light-speed misery, obviously didn't want to miss this opportunity, and decided to carry out his misery plan by filling the notepad on Ambition's computer with the character a. He had two actions!

nkxjlym opens the notepad on the Ambition's computer and types in the character a. Next, he does one of two things:

1. Spend  $v$  seconds to select all the text in the notepad, copy it, and move the cursor to the end of the text.
2. spend  $w$  seconds to press the Paste key and paste all of the text he last copied into the end of the existing text in the notepad.

nkxjlym knows that it takes at least  $k$  characters in Notepad to crash Ambition's computer. Unfortunately, before Ambition left, to prevent nkxjlym from crashing his computer, he wrote a program in his computer so that the time it takes nkxjlym to perform operation 1 is not a fixed value, but rather  $v = \text{rnd}()$ , where rnd is a random-number program that returns a new value with each call.

Ambition is coming back soon, and nkxjlym wants to know what is the shortest possible time for Ambition's computer to crash if the keys are pressed in the optimal order. Here are the C++ and py versions of this random number program.

```
int rnd()
{
    seed = (111 * seed * 7 + 13) % mod;
    return seed;
}
```

```
def rnd():
    global seed
    seed = (seed * 7 + 13) % mod
    return seed
```

### Input



The first line, an integer  $t(1 \leq t \leq 10^4)$ , represents the number of tests.

For each test, the input consists of only one line of four integers  $seed, mod, w, k(0 \leq seed \leq 10^9; 1 \leq mod, w \leq 10^9; 2 \leq k \leq 10^9)$ , the first two representing constants in the random number generator, the third representing the time required for the paste operation, and the last representing the minimum number of characters required to make the Ambition's computer die the minimum number of characters.

## Output

For each test, outputs an integer representing the minimum amount of time it took to make Ambition's computer crash.

| Sample Input 1               | Sample Output 1 |
|------------------------------|-----------------|
| 4                            | 1               |
| 0 1 1 2                      | 14              |
| 8 1 2 79                     | 3185            |
| 731 483 166 443              | 7402394725      |
| 880889741 623864658 89252520 |                 |
| 848014988                    |                 |



## G. Gem Collision

time limit per test: 1 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

You are given a string of  $n$  gems, lined up from left to right, where each gem has its own value and the  $i$ th gem is worth  $v_i$ . If you sold all the gems in the string, you could get as much money as the sum of the values of all the gems.

But the clever you realize that the energy inside the gems is unstable! For a certain interval  $[l, r] (l < r)$  of gems, you can apply a collision to the interior from the left and right ends to change the distribution of the gemstone's energy, so that the value of all the gemstones inside  $[l, r]$  becomes  $v_l \oplus v_r$ , where  $\oplus$  is a **bitwise XOR** operation. Note that the interval  $l < r$  is required to be chosen since collision is not possible when there is only one gem.

You can decide the position, order, and number of collision operations any way you like, what is the maximum value you can get?

### Input

In the first line, an integer  $t$  ( $1 \leq t \leq 10^5$ ), indicating the number of groups of tests.

For each test:

The first line inputs an integer  $n$  ( $1 \leq n \leq 10^5$ ), representing the number of gems; the second line contains  $n$  integers  $v_i$  ( $1 \leq v_i < 2^{31}$ ), representing the value of the  $i$ th gem.

It is guaranteed that the sum of  $n$  within one testcase does not exceed  $10^5$ .

### Output

For each test, output a line of integers indicating the maximum value you can get.

| Sample Input 1 | Sample Output 1 |
|----------------|-----------------|
| 7              | 6               |
| 2              | 8               |
| 1 2            | 21              |
| 2              | 56              |
| 6 2            | 186             |
| 3              | 19              |



|                                                                            |    |
|----------------------------------------------------------------------------|----|
| 3 5 2<br>4<br>2 6 4 12<br>6<br>15 1 4 8 2 16<br>2<br>9 10<br>3<br>24 25 13 | 72 |
|----------------------------------------------------------------------------|----|



## H. AGAIN! Permutation with MAX Score

time limit per test: 1 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

Define the score  $score_p$  of an arrangement  $p$  of length  $n$  to be  $\sum_{i=1}^n [(\sum_{j=1}^i p_j) = k \times p_i]$ , i.e., the number of subscripts among all subscripts of  $p$  for which the prefix sum is  $k$  times the value of the current position.

Given an integer  $n$ , you need to determine a positive integer  $k$  to find what the maximum value of the score is in all permutations of length  $n$ .

A permutation  $p$  of length  $n$  is one in which every integer from 1 to  $n$  occurs exactly once in  $p$ , e.g.,  $[3,1,2]$ ,  $[1]$  are permutations, but  $[1,1,3]$ ,  $[3,2,4]$  are not.

### Input

On the first line, an integer  $t(1 \leq t \leq 10^4)$ , representing the number of tests.

For each test, enter only one positive integer  $n(1 \leq n \leq 10^6)$ , representing the length of the arrangement.

Ensure that the sum of  $n$  within the same testcase does not exceed  $10^6$ .

### Output

For each test, the output has two lines:

In the first line, an integer  $k(1 \leq k \leq 10^{12})$ , representing the integer  $k$  you chose.

The second line, a permutation of length  $n$ , represents the highest scoring permutation.

Any answer that satisfies the requirements of the question is correct.

| Sample Input 1 | Sample Output 1 |
|----------------|-----------------|
| 6              | 1               |
| 1              | 1               |
| 2              | 1               |
| 3              | 2 1             |
| 4              | 3               |
| 5              | 3 1 2           |



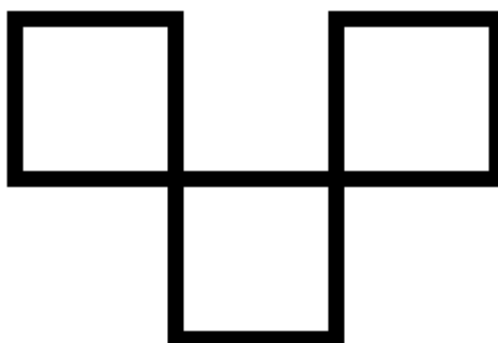
|   |                                                    |
|---|----------------------------------------------------|
| 6 | 3<br>4 2 3 1<br>3<br>4 2 3 1 5<br>3<br>2 1 5 4 6 3 |
|---|----------------------------------------------------|



## I. Board

time limit per test: 1 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

Little J has a  $n \times m$  board in front of him, and in his hand he has a number of “L” shaped puzzles as shown below:



The puzzle can be rotated or inverted to fit on the board.

He wants to ask you how many pieces can be placed on this board without the pieces overlapping each other.

### Input

On the first line, an integer  $t$  ( $1 \leq t \leq 100$ ), representing the number of tests.

For each test, input two integers  $n, m$  ( $1 \leq n, m \leq 10^6$ ), representing the length and width of the board, respectively.

### Output

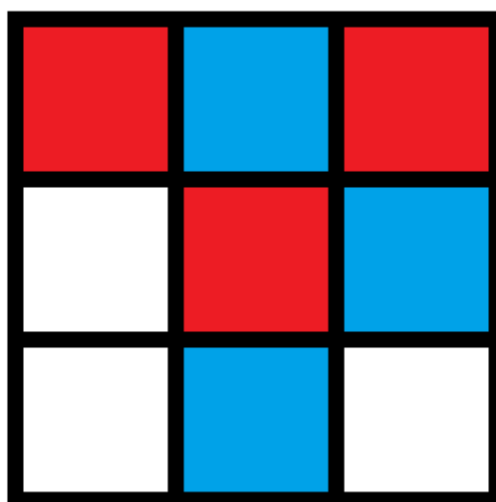
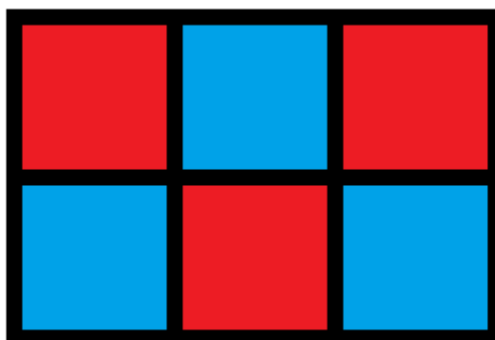
For each test, outputs a line with a positive integer representing the maximum number of pieces that can be placed in the board without the puzzles overlapping each other.

| Sample Input 1 | Sample Output 1 |
|----------------|-----------------|
| 3              | 2               |
| 2 3            | 2               |
| 3 3            | 0               |
| 1 100          |                 |



## Note

The second and third tests in the sample are placed in the following manner, respectively.





## J. Swap, Splice and Modulus

time limit per test: 2 seconds  
memory limit per test: 1024 megabytes  
input: standard input  
output: standard output

You are given a sequence of positive integers of infinite length  $a_1, a_2 \dots$  and a prime number  $p$ . And here are the following two operations on this infinite-length sequence:

- 1  $i\ j$ : for all  $k \in \mathbb{N}$ , exchange all  $a_{k \times n + i}$  and  $a_{k \times n + j}$  in the sequence.
- 2  $x$ : Find the result of taking the modulus of  $p$  after splicing <sup>†</sup> together the first  $x$  numbers of this infinite-length circular sequence.

<sup>†</sup> For two positive integers, splicing the two numbers together means joining the two numbers together like a string from left to right, resulting in a new positive integer. For example, splicing 123 and 45 together results in 12345.

### Input

In the first line, an integer  $t (1 \leq t \leq 10^4)$ , representing the number of tests.

For each test:

The first line, three integers  $n, q, p (1 \leq n \leq 3 \times 10^5; 1 \leq q \leq 3 \times 10^5; 10^9 - 10^7 \leq p \leq 10^9 + 10^7)$ , representing the length of period of the infinite sequence, the number of queries, and the modulus of operation two, respectively, ensures that  $p$  is a prime number.

The second,  $n$  integers  $a_1, \dots, a_n (1 \leq a_i \leq 10^9)$ , represent the first  $n$  numbers of the infinite sequence.

For the next consecutive  $q$  lines, each line has an existing integer  $op (1 \leq op \leq 2)$ , representing the type of operation. If  $op = 1$ , then the next two integers  $i, j (1 \leq i, j \leq n)$ , represent the exchange all  $a_{k \times n + i}$  and  $a_{k \times n + j}$  in the sequence. If  $op = 2$ , then only one number  $x (1 \leq x \leq 2 \times 10^9)$  follows, representing the query of what the result of splicing together the first  $x$  numbers of the infinite sequence module  $p$ .

It is guaranteed that for each test, the first operation type asked for  $op = 2$ .

It is guaranteed that neither  $n$  nor  $q$  will sum to more than  $3 \times 10^5$  at the same testcase.

### Output

For each query of operation type  $op = 2$ , output a line of integers representing the answer.



| Sample Input 1     | Sample Output 1 |
|--------------------|-----------------|
| 2                  | 12341           |
| 4 4 998244353      | 13              |
| 1 2 3 4            | 644986728       |
| 2 5                | 141911165       |
| 1 2 3              | 47833121        |
| 2 2                |                 |
| 2 12               |                 |
| 6 5 1000000007     |                 |
| 1 145 14 19 19 180 |                 |
| 2 6                |                 |
| 1 1 3              |                 |
| 1 5 6              |                 |
| 1 2 2              |                 |
| 2 916619           |                 |



## K. GCD of Set

time limit per test: 1 seconds

memory limit per test: 1024 megabytes

input: standard input

output: standard output

Little w likes gcd and sets so much that he came up with this problem. But little w also likes to read, so he ran off to the library after coming up with this problem. Can you help him solve this problem?

“Set gcd” means the greatest common factor of all the numbers in the set.

Formally, if the set  $A = \{a_1, a_2, \dots, a_n\}$ , then  $\gcd(A) = \gcd(a_1, a_2, \dots, a_n)$ .

In particular, if the set has only one number, then its gcd is that number, i.e.,  $\gcd(\{a\}) = a$ .

Given an irreducible set  $U$  of size  $n$ , you need to divide it into  $k$  non-empty subsets such that the sum of gcd in each subset is maximized.

### Input

In the first line, an integer  $t$  ( $1 \leq t \leq 10^4$ ), representing the number of tests.

For each test.

On the first line, enter 2 integers  $n, k$  ( $1 \leq k \leq n \leq 10^6$ ), representing the size of the set  $U$  and the number of subsets into which it is divided, respectively.

In the second line, enter  $n$  integers  $a_1, \dots, a_n$  ( $1 \leq a_i \leq 10^6$ ), with  $a_i$  different from each other, denoting the given set  $U$ .

It is guaranteed that the sum of the maximum values of  $a_i$  within the same test does not exceed  $10^6$ .

### Output

Outputs an integer representing the maximum value of the sum of gcd for each subset.

| Sample Input 1 | Sample Output 1 |
|----------------|-----------------|
| 2              | 8               |
| 3 2            | 38              |
| 3 6 5          |                 |
| 5 4            |                 |



|              |  |
|--------------|--|
| 13 11 10 9 5 |  |
|--------------|--|