Irish Collegiate Programming Competition 2013

Problem Set

University College Cork ACM Student Chapter

April 6, 2013

Instructions

Rules

- All mobile phones, laptops and other electronic devices must be powered off and stowed away for the duration of the contest.
- The only networked resource that teams are permitted to access is the submission system.
- If a team discovers an ambiguity or error in a problem statement they should tell an organiser. If the organisers agree that an ambiguity or error exists, a clarification will be issued to all teams.
- No multi-threading is allowed, and no sub-processes.
- No file input/output is allowed. All input to your program will be provided via standard input (stdin) and all output should be printed to standard output (stdout). Examples of how to do this in each of the languages is provided in the resources section of the submission site.

Testing and Scoring

- A solution will be tested against 10 test cases. Output from your program should be terminated by a new line and should not contain any additional whitespace at the beginning or end of lines.
- A solution will be accepted if it compiles on the test environment and produces the correct output for the sample inputs given in the question.
- If a solution is submitted while an earlier solution to the same problem is still in the queue, the earlier submission will not be tested. The earlier submission will be marked with the message: "Old, not going to be tested".
- All accepted problems will be scored out of 10 points, one for each test case that returns the correct output.
- You will not be told the points scored until the end of the contest.

- In the event of a tie, the winning team will be the one with the highest points on problem 5, if both teams score the same on problem 5, then the one with the highest points on problem 4 and so on. If a tie still exists, the team who's last scoring submission was submitted earliest will be the winner.

Submission Instructions

- The submission system URL is: `http://4c245.ucc.ie:8080/`
- Your password will be provided by the organisers. Please notify an organiser if you unable to log in.
- Submissions should consist of a single source file, **not a compiled executable**.
- To submit, click the "Submit a Solution" link, complete the submission form, upload your source file, and click save.
- Your submission should now be listed in your submission queue.
- Submission input is read from standard input and submission output should be written to standard output.
- To teams submitting Java solutions, be aware the testing script will be renaming your file to `Pn.java` (where n is the problem number) and as such the main class for problem 1 should be defined as:

```
public class P1 {
...
}
```

and will be called with:

```
java P1 < input-file
```

1 Sum Free

A set of integers, $\mathcal{X}$, is said to be **sum-free** if the sum of any two integers in $\mathcal{X}$ is not in $\mathcal{X}$. More formally:

$$\forall a, b \in \mathcal{X} : (a + b) \notin \mathcal{X}$$

Given a set of integers as input determine whether it is sum-free.

Input

The first line contains an integer N, $2 \leq N \leq 1000$, the number of integers in the set. The second line contains N space-separated positive integers.

Output

The output of your program should be 1 if the set is sum-free or 0 otherwise. The answer 1 or 0 should immediately be followed by a single newline character.

Sample Input 1

```
5
4 5 15 2 8
```

Sample Output 1

```
1
```

Sample Input 2

```
8
12 3 10 27 36 29 15 8
```

Sample Output 2

```
0
```

2 Primed Positioned Primes

A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. The first 10 prime numbers are:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29$$

The first order super prime list is the primes positioned at prime indices in the list of all primes. For example, the first order prime list from the ten primes above is: $3, 5, 11, 17$ since the value 3 is at index 2, 5 is at index 3, 11 is at index 5, and so on. Note that lists are 1-indexed.

The second order super prime list is the primes positioned at prime indices in the first order super prime list. The second order super prime list given the first order prime list above is: $5, 11$.

In general, the super prime list of order n is the list of primes positioned at prime indices in the super prime list of order $n - 1$. We denote the list of super primes of order n as S_n. The base case super prime list S_0 is an initial list of primes with values between a lower bound a and an upper bound b, inclusive. Given a lower bound a, an upper bound b and an order n your task is to find the list of super primes S_n.

Input

Input consists of a single line, containing 3 space-separated integers a, b, n.
$2 \leq a \leq b \leq 3000000$ and $0 \leq n \leq 10$.

Output

The output of your program should consist of the list of space-separated super primes of order n, where the base case prime list S_0 containts all primes between a and b inclusive. The list should be sorted increasingly. The list should never be empty and should be immediately followed by a new line character.

Sample Input 1	**Sample Output 1**
`100 110 0`	`101 103 107 109`

Sample Input 2	**Sample Output 2**
`2 10 2`	`5`

3 Motorcycle Meeting Madness

In Ireland there exists a clandestine network of motorcycle gangs. These gangs like to convene and share their motorcycle gang ideas. Unfortunately, fuel is expensive and even the motorcycle gangs are affected by the tough economic climate. The motorcycle gangs need your help to save money. Your task, should you choose to accept it, is to identify a common location where the gangs can convene such that their total fuel bill will be minimized.

Given a set of locations L, a set of links between these locations, and a set of motorcycle gangs G of varying sizes at some locations in L, find the location which has the lowest fuel cost for all groups to travel to. All members of a gang start in the same location and they each travel on their own motorcycle and all members of a group must travel the same route. The cost for a group to travel a route is the number of gang members times the fuel cost for that route.

Input

First line of input contains 3 space-separated integers n, e and g, with $2 \leq n \leq 150$, $n \leq e \leq 2500$, and $1 \leq g \leq n$. The integer n is the number of locations that the gangs are aware of. To keep the locations of their meetings secret, the gangs have given each location an integer identifier starting from zero. e is the number of links between locations, and g is the number of groups.

The next e lines each contain 3 space-separated integers a, b and f, with $0 \leq a, b < n$, $a \neq b$, and $0 < f \leq 10$. This means there is a bi-directional link between secret locations a and b with a fuel cost of f.

The following g lines each contain 2 space-separated integers c, d, with $0 \leq c < n$, and $0 < d \leq 50$. This describes a motor cycle gang at a secret location c of size d.

Output

Your program should output a line containing two space-separated integers, i and c. i is the identifier of the secret location that all gangs should travel to, which has the lowest cost to travel to. c is the total cost for all groups to travel to location i. This line should be followed by a single new line character.

Sample Input 1

```
5 8 4
0 1 2
0 4 2
0 2 1
1 2 1
1 3 3
2 3 1
2 4 1
3 4 2
0 3
1 3
3 3
4 4
```

Sample Output 1

```
2 13
```

Sample Input 2

```
6 7 5
0 1 3
0 2 1
1 5 4
2 3 1
2 4 2
3 5 6
4 5 3
1 6
2 1
3 3
4 2
5 4
```

Sample Output 2

```
1 47
```

4 Ghostbusters and Ghosts Gun Grapple

```
"There's something very important I forgot to tell you! Don't
cross the streams ... It would be bad ... [...] and every
molecule in your body exploding at the speed of light."
     |Egon Spengler to Harold Ramis on crossing proton streams.
```

As they always do, a group of n ghost busters is battling a group of n ghosts. The ghost busters are armed with proton guns that shoots a stream in a straight line and terminates when the aimed ghost is hit. The ghost busters agreed on the following strategy : They will form n ghost buster-ghost pairs and shoot simultaneously at their chosen ghost. As the above quote reminds you, it is very dangerous to let the proton beams cross as it will generate an explosion. Every living being within the radius of the explosion will be vaporized out of existence. Your task is thus, given the position of the ghosts and the ghost busters, the configuration of the matching and the radius of the explosion(s), is to find how many ghost busters survive the battle.

We will assume that the positions of ghosts and ghost busters are fixed points in the plane and that no two beams are strictly overlapping.

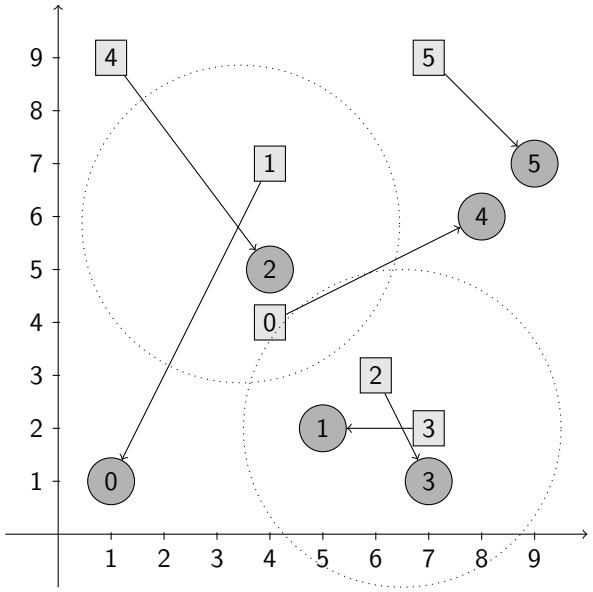

Figure 1: 6 ghost busters chasing 6 ghosts. The outcome is two explosions of radius 3, killing 4 ghost busters (0, 1, 2, 3). Hence, the expected answer to this particular setup is 2. The input corresponding to this scenario is given as sample input 2.

Input

The first line of input consists of two integers n and r, the number of ghosts and the radius of the explosions respectively. $1 \leq n \leq 150$ and $0 \leq r \leq 200$.

Each of the next n lines contains two integers x and y representing the positions of the ghosts 0 up to $n-1$.

The following n lines contains two integers x and y representing the poitions of the ghost busters 0 up to $n-1$. For all positions $0 \leq x, y \leq 1000$.

The next n lines contains two integers a and b each describing the ghost-ghostbusters matching. a is the index of a ghost and b is the index of its paired ghostbuster.

Output

The output of your program should consist of a single integer corresponding to the number of living ghostbusters after the fight, immediately followed by a new line.

Sample Input 1

```
4 5
10 8
2 6
4 0
0 8
8 9
2 4
4 7
8 0
0 3
1 0
2 2
3 1
```

Sample Output 1

```
2
```

Sample Input 2

```
6 3
1 1
5 2
4 5
7 1
3 1
9 7
4 4
4 7
6 3
7 2
1 9
7 9
0 1
2 4
4 0
3 2
1 3
5 5
```

Sample Output 2

```
2
```

5 Peculiar Pizza Partitioning Problem

Pete's Polygon Pizza Parlour is proud of its peculiar polygon pizzas and particularly its precise pizza partitioning process. Carefully cutting customers cooked chow according to $CumulativeCrustRatio^{TM}$.

Pizzas are cooked in the shape of polygons, rather than circles. Then pizzas are cut from one corner of the polygon to another, ensuring that cuts never overlap, until the pizza is cut into triangles. However Pete knows that any sensible person likes pizza crust (delicious, crispy outer edge) and wants a good amount of crust on each slice. So, he devised his $CumulativeCrustRatio^{TM}$. $CumulativeCrustRatio^{TM}$ is the sum of the crust to non-crust ratio for each of the pizza slices. When slicing the pizza, he chooses to slice it in the way that has the highest $CumulativeCrustRatio^{TM}$ score.

Given a series of vertices V, describing the polygon pizza P, the goal is to find a triangulation of the polygon pizza, which has the highest $CumulativeCrustRatio^{TM}$. A triangulation is formed by completely partitioning the polygon into non-overlapping triangles described by vertices in V. Formally, $CumulativeCrustRatio^{TM}$ is defined for the triangulation T as:

$$\sum_{t \in T} \frac{external(t)}{internal(t)}$$

where $external(t)$ is the length of the triangle t's edges which are also external edges of the polygon P, and $internal(t)$ is the length of the triangle t's edges which are the internal partitions of P using the triangulation T.

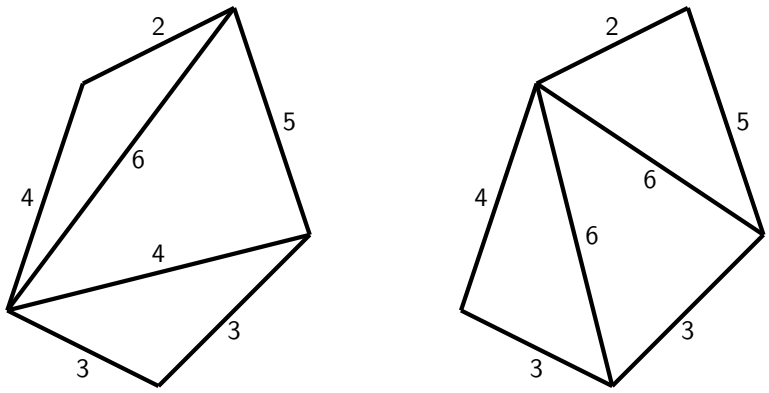

Figure 2: Two possible triangulations of a polygon. The first triangulation has a $CumulativeCrustRatio^{TM}$ of 3 (1 + .5 + 1.5) while the second has 3.58 (1.666667 + .25 + 1.666667)

Input

The first line of input consists of a single integer N, $4 \leq N \leq 100$, the number vertices of the pizza polygon. The next N lines each contain two integers X and Y, $0 \leq X, Y \leq 50000000$, the coordinates of a vertices of the polygon.

Output

The output of your program should be the score of the best partitioning of the pizza. The score should be rounded to 6 decimal places. The score should immediately be followed by a single newline character.

Sample Input 1

```
4
5 3
2 1
6 2
3 3
```

Sample Output 1

```
3.090616
```

Sample Input 2

```
5
4 3
6 5
3 8
5 7
2 4
```

Sample Output 2

```
2.994481
```