UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Course Title: Data Structure & Algorithm Lab Lab II **Course Code: CSE2218**

Trimester : Summer 23 **Section: C** **Credit Hours: 1.0**

ASSIGNMENT 02: Greedy & DP

Q1: Burst The Balloons

There are some spherical balloons taped onto a flat wall that represents the XY-plane. The balloons are represented as a 2D integer array of points where `points[i] = [`x_{start}`, `x_{end}`]` denotes a balloon whose horizontal diameter stretches between x_{start} and x_{end}. You do not know the exact y-coordinates of the balloons.

Arrows can be shot up directly vertically (in the positive y-direction) from different points along the x-axis. A balloon with x_{start} and x_{end} is burst by an arrow shot at x if $x_{start} <= x <= x_{end}$. There is no limit to the number of arrows that can be shot. A shot arrow keeps traveling up infinitely, bursting any balloons in its path.
Given the array points, return the *minimum number of arrows* that must be shot to burst all balloons.

Example 1:

Input: [[10,16],[2,8],[1,6],[7,12]]	**Output:** 2

Explanation: The balloons can be burst by 2 arrows:

- Shoot an arrow at x = 6, bursting the balloons [2,8] and [1,6].

- Shoot an arrow at x = 11, bursting the balloons [10,16] and [7,12].

Example 2:

Input: [[1,2],[3,4],[5,6],[7,8]]	**Output:** 4

Explanation: One arrow needs to be shot for each balloon for a total of 4 arrows.

UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Q2: Shooting Game

Samu is playing a shooting game in play station. There are two apples to aim in this shooting game. Hitting first apple will provide her **X** points and hitting second apple will provide her **Y** points. And if she misses the apple she chose to hit, she won't get any point. Now she is having **N** coins and each shoot will cost her **1** coin and she needs to score at least **W** points to win the game.

Samu don't like to lose at any cost. At each turn, she has two choices. The choices include:-

- Hitting first apple with probability **P1** percent. However, she might miss it with probability **(1-P1)** percentage.
- Hitting second apple with probability **P2** percent. However, she might miss it with probability **(1-P2)** percentage.

She would like to know what is the maximal expected probability (as a percentage b/w 0 and 100) of winning the shooting game.

Input Format:

- First line contains the number of test cases **T**.
- Each test case consists of six space separated integers of the form **X Y N W P1 P2** as described in the statement.

Output Format: For each test case, print the result as described above in a separate line.

Note: Choosing to hit any apple is entirely her choice. Both are independent events meaning **P1** + **P2** may/may not exceed 100. Output must contain 6 digits after decimal.

Example:

Input:	**Output:**
1 2 3 2 5 50 25	12.500000

Explanation: Samu is getting **2** points from shooting first apple and **3** points from shooting second Apple.

She had **2** chances to shoot and she need to score atleast **5** points so anyhow she need to shoot Apple 1 in one shoot and Apple 2 in another shoot , if she wants to win.

The maximum probability of winning is **0.5 * 0.25 = 0.125 = 12.5%**

Constraints:

$1 \leq T \leq 10$
$1 \leq X,Y \leq 10$
$1 \leq N,W \leq 10^3$
$0 \leq P1,P2 \leq 100$

UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Q3: Maximize Marks

An exam consists of **N** questions. The marks of the N questions are $\mathbf{m_1, m_2, m_3, .. m_N}$ respectively. Jam is giving the exam and he wants to maximise his number of marks. However he takes some time to solve each question. The time taken by him to solve the questions are $\mathbf{t_1, t_2, t_3, .. t_N}$ respectively. The exams lasts for a total of time **T**.

But Jam's teacher is very smart and she knows that Jam will find out a way to get maximum marks. So, to confuse Jam, she also puts up a bonus offer for him - The offer is that Jam can select a question for which he can double the marks awarded for that question. Now, Jam is indeed confused. Help him find out the maximum number of marks he can gain.

Input

- The first line contains a single integer **N** that represents the number of questions.
- Second line contains a single integer **T**, the total time for which the exam takes place.
- Third line contains N space-separated integers $\mathbf{m_1, m_2, m_3, ... m_N}$, where $\mathbf{m_i}$ represents marks assigned to the $\mathbf{i^{th}}$ question.
- Fourth line contains N space-separated integers $\mathbf{t_1, t_2, t_3, ... t_N}$, where $\mathbf{t_i}$ represents time taken to solve the $\mathbf{i^{th}}$ question.

Output

Output a single integer, that is the maximum number of marks Jam can achieve.

Example

Input:	**Output:**
3 10 1 2 3 4 3 4	8

Constraints

- 1<=**N**<=1000
- 1<=**T**<=10000
- 1<=$\mathbf{m_i}$<=100000
- 1<=$\mathbf{t_i}$<=10000

Q4: Efficient Delivery

Sergey works as a delivery boy. Every day he has a number of deliveries to be completed.Sergey lives in Bytetown - a city, consisting of **N** junctions, and **M** roads connecting these junctions. More precisely, the $\mathbf{i}^{th}$ road connects the junctions with the numbers $\mathbf{X_i}$ and $\mathbf{Y_i}$. Each of these roads has its own (positive) length $\mathbf{Z_i}$, and no road connects a junction to itself.

There are **K** delivery orders for Sergey.Each order is denoted with **four** integers: $\mathbf{A_j}$, $\mathbf{B_j}$, $\mathbf{V_j}$ and $\mathbf{C_j}$. It means that for the $\mathbf{j}^{th}$ order Sergey has to pick a parcel of weight $\mathbf{V_j}$ at the junction numbered $\mathbf{A_j}$ and deliver it to the junction numbered $\mathbf{B_j}$. After the parcel is delivered, Sergey gets $\mathbf{C_j}$ burles of reward. Sergey can't complete the same order twice or more times. So, the business day for Sergey looks as follows: he remembers the whole list of orders and starts at his initial junction numbered **S**. Then, he travel the streets of the city via his bike. If Sergey visits a junction, where there is one or more orders available he can either take some of the orders and load the corresponding parcels on his bike, or not take anything now but do it later, if he appears at the same junction again. When Sergey visits a junction which is the destination for some of the parcels that are currently with him, he can complete these orders and unload the corresponding parcels.

Sadly, Sergey is not infinitely strong. Though he can carry any number of parcels at once, if the total weight of the parcels exceeds **W** Sergey won't be able to carry them on his bike. Sergey also can't take a parcel and not deliver it because there will be complains, hurting Sergey's reputation.
The bike has enough fuel to travel only for **D** units of distance. It is not necessary for Sergey to get back to his initial junction because he has a lot of friends in Bytetown who will drop him back after the party (Sorry, you're not invited). It is clear that under the given constraints, Sergey might be not able to deliver all the parcels. So your task is to plan the delivery for Sergey in such a way that:

- He moves only by streets
- The total travelled distance doesn't exceed **D**.
- There are no undelivered parcels with Sergey by the end of the delivery. In other words, all the parcels that Sergey has picked up should be delivered.
- At any time, Sergey does not carry more than **W** units of weight.
- The most important: the profit Sergey gets by the end of the day is maximized. You don't have to find the optimal delivery, but the more profit there is, the more are the points you'll get.

Input

- The first line contains two space separated integers **N** and **M** denoting the number of junctions and the number of streets.
- Each of the following **M** lines contain three space separated integers $\mathbf{X_i}$, $\mathbf{Y_i}$ and $\mathbf{Z_i}$, denoting that the $\mathbf{i}^{th}$ street of length $\mathbf{Z_i}$ connects the $\mathbf{X_i}^{th}$ and the $\mathbf{Y_i}^{th}$ junctions. All the streets are **bidirectional**.
- The next line contains a single integer **K** denoting the number of orders.
- Each of the following **K** lines contain four space-separated integers $\mathbf{A_j}$, $\mathbf{B_j}$, $\mathbf{V_j}$ and $\mathbf{C_j}$, denoting that for the $\mathbf{j}^{th}$ order, the parcel should be picked up at the $\mathbf{A_j}^{th}$ junction, left at the $\mathbf{B_j}^{th}$ junction, has the weight of $\mathbf{V_j}$, and when delivered, gives $\mathbf{C_j}$ burles of profit.

- The following line contains three space-separated integers **S**, **D** and **W** denoting the starting junction, the maximal distance that Sergey can travel and the maximum weight he can carry at any time, respectively.

Output

The first line of output should contain a single integer **Op**, denoting the number of operations. Each of the following **Op** lines should contain the operation code and the parameter. The following are the descriptions of the operation codes and the meanings of the parameters:

- 0 **P**. Here **0** is the operation code and **P** is the parameter. The meaning is that Sergey should move from his current junction to the adjacent junction **P**. In case there is no **direct one-street** connection between Sergey's current position and the junction **P**, you will receive a *wrong answer* verdict. In case Sergey is unable to reach the junction **P** because the amount of fuel is too low, you will also get *wrong answer* verdict.
- 1 **P**. Here **1** is the operation code and **P** is the parameter. The meaning is that Sergey should takes the order numbered **P** in the orders' list. In case this order was already taken or Sergey is not in the junction which is the pick-up junction for the order, you will get a *wrong answer* verdict. In case the weight that Sergey will have to carry exceeds **W** after taking this order, you'll also get a *wrong answer* verdict.
- 2 **P**. Here **2** is the operation code and **P** is the parameter. The meaning is that Sergey should deliver the parcel for the order number **P** in 1-based orders' list. In case Sergey doesn't carry the parcel from this order, you'll receive *wrong answer* verdict.

You can also get a *wrong answer* if your output doesn't satisfy the format above or some parcels picked up by Sergey are left undelivered. If all the requirements are satisfied, you'll receive an *accepted* verdict ;).

Input:	Output:
5 5 1 2 1 2 3 2 1 4 1 4 5 1 5 2 3 3 1 4 5 10 2 5 6 15 4 1 10 10 1 5 12	7 1 1 0 2 1 2 0 5 2 2 0 4 2 1

Explanation:

Sergey starts at the junction numbered the **1**st. Before moving on, he takes the order numbered the **1**st. Then he moves to the **2**nd junction and there he takes the **2**nd order. Then, Sergey moves to the **5**th junction, where he delivers the parcel from the **2**nd order. Finally, he moves to the **4**th junction, where he delivers the parcel from the **1**th order. The total profit that Sergey receives from this delivery is **(10 + 15) = 25** burles. Hence, this answer will get the score of **25**.