Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Art Transaction

`art.c`, `art.cpp`, `Art.java`, `art.py`

The largest picture stolen by the gangsters from the State Gallery is to be evaluated by an expert before it is sold on the black market. The composition of the picture clearly follows mainstream tendencies in contemporary AI art. The picture can be viewed as a grid of square cells where each cell is either empty or non-empty. A non-empty cell contains exactly one of these objects: a sun, a house, a chupacabra, a left slope, a right slope, a bird, a drake, or a grill. Two cells are adjacent if they share an edge in the grid. A connection between two cells is a sequence of cells which contains both given cells and in which each two consecutive cells are adjacent in the picture. An area is a set of cells. It is connected if there is a connection between any two cells in the area.

The value of the picture is the total of all values generated by all rules listed below.

Suns:
A non-empty cell, not containing a sun, is illuminated by a sun if there is no other object on a straight line between the cell and another cell with a sun. The straight line may be horizontal, vertical or diagonal (both diagonal directions apply). When a cell is illuminated by multiple suns it is still counted only once. A sun cannot illuminate itself. Each illuminated cell generates 100$.

Biggest bird:
A drake is a bird. A flock of birds is maximal connected area of cells each of which contains a bird. The width of a flock is the length of a maximum contiguous sequence of cells on one line in a flock. A flock of birds generates 500$ $\cdot$ (*width of the flock*).

Flock perimeter:
Each flock of birds (see previous rule) generates value 60$ $\cdot$ (*flock perimeter*). Flock perimeter is the total number of edges each of which separates a flock cell from a cell which does not contain a bird or from the outside of the picture.

House view up:
When an empty cell is located above a cell containing a house, it is in the same column, and there is no non-empty field between the empty cell and the house, the empty cell generates 10$.

3×3 blocks:
Each unique 3×3 block (possibly overlapping) generates 1$.

Animals I:
Each edge between a cell with an animal and an empty cell generates 15$.

Freedom:
Cell X is a freedom cell if it is either adjacent to the picture border or there is a connection between a cell adjacent to the picture border and a cell adjacent to X and the connection contains only empty cells. Each non-empty freedom cell generates 7$.

Chupacabra:
A drake is a bird. Each bird which can be reached by a chupacabra performing one chess knight

move in the grid generates 200$.

Peaks:
A pair of adjacent cells with characters "/\" (left slope and right slope, in this order) on one line is a peak. Peak summit is an imaginary point in the middle of the segment connecting the uppermost points of both symbols in the peak. Peak value is equal to the maximum Manhattan distance from the peak summit to the summit of another peak in the grid. Note that peak value is an integer. Each peak P generates 50$ $\cdot$ (*P value*) when there are at least two peaks in the picture. Otherwise, a single peak in the picture generates 0$.

Drake/grill:
Each cell with a drake and with at least one adjacent cell with a grill generates 500$.

Minimum frequency:
Frequency of an object X is the number of the objects (including X) of the same type as X in the entire picture. Each single object which frequency is minimum among all objects in the picture generates 10$.

Empty fields:
Each empty field generates 1$.

Animals II:
All animals in the picture together generate single value 1$ $\cdot$ (*number of chupacebras*) $\cdot$ (*number of birds which are not drakes*) $\cdot$ (*number of drakes*).

House view down:
When an empty field is located above a house, is in the same column, and there is no non-empty field between the empty field and the house, the empty field generates 5$.

Grill/drake:
Each cell with a grill and with at least one adjacent cell with a drake generates 50$.

Houses and grills:
Houses and grills in the picture generate 3$ $\cdot$ min { *number of all houses*, *number of all grills* }.

Input Specification

The first input line contains one integer N ($1 \leq N \leq 50$), the number of the rows and the columns in the picture. Next, there are N lines, each specifies one line in the picture. One character on the line represents one cell and its contents.

The interpretation of particular characters in the picture representation follows:
empty cell " " (space), sun "`*`", house "`^`", chupacabra "`!`", left slope "`/`", right slope "`\`", bird "`v`", drake "`D`", grill "`G`".

Output Specification

Print one integer, the value of the input picture.

Sample Input 1

```
9
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
 *^!/\vDG
```

Output for Sample Input 1

```
12672
```

Sample Input 2

```
3
!
  v
 D
```

Output for Sample Input 2

```
2059
```

It is highly recommended to use sample inputs which are attached to the problem in DOMJudge.

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Bank Robbery

`bank.c`, `bank.cpp`, `Bank.java`, `bank.py`

Each day, robbers plan to rob exactly one bank in a region. Due to undercover informers' work, the detectives get to know which particular bank is being targeted by the robbers on each day just before 6 AM. The presence of a single detective in a bank is enough to deter the robbers, so the detectives want to plan their positions intelligently. Robbing is planned to happen during the daylight after 8 AM, when the banks open, and is always successful when there is no detective in the bank. Unfortunately, though, the number of detectives is not very big so there may not be enough of them to keep the banks safe. To ensure they are effective, there can be at most one detective in any bank at any time. A detective can leave a bank and travel to another bank only between 6 AM and 8 AM.

Given enough days, any detective would be able to move from any of the banks to any other via the 2-hour transits. Due to the travel restrictions (mainly time), they are not able to move freely but are restricted to move only between banks that are close to each other. The region is quite specific because there is a minimum number of bank pairs that are close that conform to the above restrictions. Additionally, no bank is close to exactly two other banks.

Now, there is a quest for a computer simulation: determining if the robbers can succeed in at least one robbery during one year. The simulation is run against the preprogrammed judge as a kind of computer game, with strong consequences in real life. In the game, the attacker represents the robbers, the defender manages the detectives.

In the beginning, the simulation is given the system of connections between close banks and the number of detectives available. Then, the simulation chooses whether it plays as an attacker or as a defender. The judge automatically adopts the opposite role. Next, the defender places the given number of detectives in banks according to its own choice.

Next, the game proceeds in turns. In one turn, the attacker announces a bank and then the defender moves each detective over at most one connection. The defender's aim is to choose the movements in such a way the announced bank is occupied by a detective at the end of the turn. If there is no detective in the announced bank after all movements in the turn, the attacker immediately wins the game. Otherwise, the defender defends the turn and the next turn ensues. If the defender can successfully defend for one year (365 turns), they win the game. During the game, the location of each detective is known to both the attacker and the defender.

The goal of the simulation you have to write is to choose a role smartly so that you are able to win the game.

Input Specification

The first input line contains values B and D ($4 \leq B \leq 100, 0 \leq D \leq B$), the number of banks and the number of detectives. The banks are labeled by integers $0, 1, \ldots, B-1$. Each of the next $B-1$ lines contains a pair of integers b_i and c_i ($0 \leq b_i, c_i \leq B-1$) representing a connection between two close banks b_i and c_i.

Output Specification

After reading the input, if you choose to attack, then prints a line with string `ATTACK`. Otherwise, print a line with string `DEFEND` and on the next line it print D different indices, in arbitrary order, of all the banks where a detective is initially located. The remainder of the exchange happens interactively.

Interactive Mode

The simulation is evaluated in so-called *interactive* mode, which means that the input received depends on the output produced so far. The output of the judge is the input of the simulation and vice versa. If you have no previous experience with such problems, just do not be afraid — you are still reading from the standard input and printing to the standard output. There are just a few things to pay attention to.

After printing each response to the input from the judge, the simulation has to flush the output buffer. For example, it may use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, or `stdout.flush()` in Python. Also, it should never try to read more data than what is guaranteed to be ready in the input, because that may cause it to hang indefinitely. In particular, be careful to *not* invoke anything like `scanf("%d ")` or `scanf("%d\n")`, as such formats try to scan forward for any further whitespace. Instead, use just `scanf("%d")` without trailing whitespace.

After you choose a role and print a corresponding output, the following exchange is repeated for up to 365 times: The attacker outputs the bank index $0 \leq v \leq B-1$ which they choose to attack. Then, the defender outputs an integer k and then k pairs b_i, c_i ($0 \leq b_i, c_i \leq B-1$), each of which represents a detective move from bank b_i to bank c_i. The detectives may only move along connections between close banks. The attacker ends the sequence of attacks by outputting -1. The defender may give up by ending the program.

In this exchange, the output of one player is always the input of the other player.

Sample Input 1

```
4 3
0 1
0 2
0 3

2

3

-1
```

Output for Sample Input 1

```
DEFEND
0 1 2

0

2 1 0 0 3
```

Sample Input 2	Output for Sample Input 2
4 1	
0 1	
0 2	
0 3	
	ATTACK
0	
	1
1 0 1	
	2

For clarity, the above data are interleaved to illustrate the order of interaction between the simulation and the judge. Note that there will be no empty lines in real data and there must not be any empty lines in the simulation output.

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Pizzo Collectors

`collectors.c`, `collectors.cpp`, `Collectors.java`, `collectors.py`

Lavish Circle (LC) is the fashionable circular avenue in the residential area of the town. Houses on LC are exceptionally expensive and some of them are currently empty. LC is under heavy control of the town mafia who wants to populate the empty houses with new owners loyal to the mafia. When LC is completely populated, each house owner will live in one house on LC. LC is a circular avenue of houses numbered from 1 to N, that is for $i < N$, i-th and $i + 1$-th houses are neighboring and also houses N and 1 are neighboring.

The house owners, both the existing ones and the new ones, fall into few categories according to the sum they can pay the mafia monthly for protecting them. The money is called pizzo and it is typically collected by a person called a pizzo collector (PC). The mafia employs a group of them.

The job of a PC is to go around entire LC exactly once in a month and collect pizzo from the selected houses on the journey. All selected houses on a journey of a PC must have owners of the same pizzo category. The journey must also start and end at the same house, it is a check PC completed the journey correctly. The pizzo is collected from this house only once, at the beginning or at the end of the journey. During his journey, a PC always moves forward by a fixed number of houses, until the PC arrives again to the starting house. That is, the number of houses a PC skips on each move is a non-negative integer d, which remains constant during the entire journey of this PC. It must hold that $(d + 1)$ divides N evenly.

The mafia wants to employ as many PCs as possible. Of course, employing a number of PCs means that some owners quite probably have to pay pizzo more than once in a month, but the mafia does not care... Unfortunately, there is a complication. PCs are peaceful citizens and they do not tend to shoot at each other under normal conditions. However, when two PCs find out that at their respective collection journeys they visit the same set of houses, and it does not matter in which order they visit the houses, the collectors tend to shoot each other and thus attract the police, which is a behavior the mafia wants to avoid at any cost. So, no two collectors who may shoot each other can be employed simultaneously.

The total value of all collected pizzo depends also on the categories of the owners of the newly populated houses. The mafia decides on the category of each new house owner. Obviously the mafia wants to maximize their income. You have been hired as an analyst to find the maximum possible total value of all collected pizzo in one month, when LC gets completely and suitably populated. The mafia is going to decide on the pizzo category of each new house owner based on your recommendations. The number of houses on LC is a non-negative integer power of a prime number.

Input Specification

The first line contains integer N ($1 \leq N \leq 10^5$), a non-negative integer power of some prime number p. The second row contains string S of length N, which consists of only capital letters of English alphabet and the character "?". The characters of the string represent the houses on LC in the order they appear on LC. The "?" character represents a currently empty house, each of the other characters represent the pizzo category of the house owner.

The next line contains integer k ($1 \leq k \leq 26$), the number of different pizzo categories. Each of the next k lines contains an integer pair c_i a v_i, where c_i is a capital English character and $1 \leq v_i \leq 10^6$ is the money value which is paid by a house owner of pizzo category c_i in one visit of a PC to the house.

It is guaranteed that for every category that appears in S, there is a pair c_i and v_i which defines its money value which is paid on the PC's visit.

Output Specification

Print the maximum possible total value of all collected pizzo in one month, when LC is completely populated.

Sample Input 1

```
4
A?A?
2
A 10
B 25
```

Output for Sample Input 1

```
140
```

Sample Input 2

```
4
A??A
2
A 10
B 25
```

Output for Sample Input 2

```
120
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Excavation

`excavation.c`, `excavation.cpp`, `Excavation.java`, `excavation.py`

The police investigation revealed the gangsters deployed several radioactive stones under the city to poison underground waters. The exact positions of radioactive stones were found, but due to nature of radioactivity, it is a difficult task to remove the stones safely. The government of the city thus decided to use shielded excavators to retrieve stones from the ground.

The city shape is a square grid. City services have several excavator types available – Reepadlo, Qrtech, Bugger, Namakatschenko, and Kopatsch. Each of them has different specifications and movement pattern. Excavators may move either as a Rook, a Queen, a Bishop, a kNight, or as a King in the game of chess, respectively (see images for movement illustration). Due to compatibility issues only a single type of excavators can be deployed.

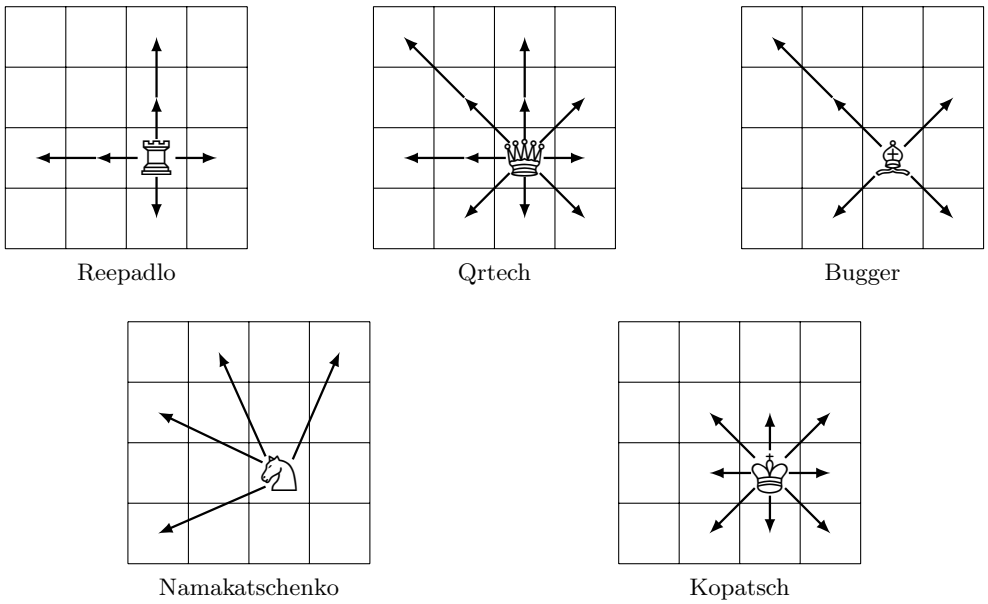


Figure 1: Illustration of specific excavator movement patterns.

There is at most one radioactive stone on each tile of the grid. At the beginning of the excavation, there is one excavator at the position of each radioactive stone and it immediately retrieves the stone from the ground. The next steps of the operation are executed to follow strict radiation handling safety protocol. At each step only one excavator can execute a single move and it can execute it only if the move brings the excavator to a position of another excavator. Excavators of types Reepadlo, Qrtech, Bugger may skip over other excavators during a move over multiple grid tiles, i.e. they do not have to end the move on the position of the first encountered excavator. After excavator A arrives to the position of excavator B, B takes its load and A is removed from the operation to be cleaned of radiation.

The operation finishes succesfully if in the end there is a single excavator remaining. It is possible the operation can not be successfuly finished.

Your task is to determine whether the operation can be finished succesfully. If it can, print also the excavators' moves leading to the solution.

Input Specification

The first line of input contains an integer N ($1 \leq N \leq 100$), determining the size of the city, and a single character determining the excavator type to be deployed ("`R`" – Reepadlo, "`Q`" – Qrtech, "`B`" – Bugger, "`N`" – Namakatschenko, "`K`" – Kopatsch).

After that follow N lines describing the initial positions of the excavators in the city. Each line contains N characters, where each character is either the excavator type or "`.`" for empty field. There is always at least one excavator deployed in the city.

Output Specification

On the first line print either "`YES`", if the operation for the given configuration can be finished successfully, and "`NO`" otherwise. If the operation can be finished successfully, print also lines describing moves of excavators in the same order they were executed during the excavation, if there were any. i-th such line describes a single step and contains four space separated integers X, Y, W, Z ($1 \leq X, Y, W, Z \leq N$), indicating an excavator moves from position (X, Y) to position (W, Z) in step i. A position (X, Y) describes the position on row X (numbered from top to bottom) and in column Y (numbered from left to right).

Sample Input 1

```
2 K
K.
KK
```

Output for Sample Input 1

```
YES
2 2 2 1
2 1 1 1
```

Sample Input 2

```
3 B
B..
B..
..B
```

Output for Sample Input 2

```
NO
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Tobacco Growing

`growing.c`, `growing.cpp`, `Growing.java`, `growing.py`

In the city, there is a beautiful park full of blossoming flowers. This park is very popular among citizens and especially families with children are common visitors. The gangsters want to extend their criminal activities to this area, mainly in the tobacco production. They decided to secretly plant tobacco seeds in the park and let it grow right under policemen noses. When the time is right, they plan to harvest as much tobacco as possible in a single operation and smuggle it over borders.

The park consists of flower tiles with coordinates ranging from -10^6 to 10^6 in either direction. A tile on coordinates (X, Y) neighbors with tiles on coordinates $(X+1, Y)$, $(X-1, Y)$, $(X, Y+1)$ and $(X, Y-1)$ (within coordinates range). On the first day, the gangsters may cut the flowers on any of the tiles. Also only on the first day, if a tile is cut, the gangsters may either plant a tobacco on it, or let the tile contain only grass. In the beginning, the tobacco quantity on a tile with tobacco is 1 and the tobacco quantity on a tile with grass or on a tile with flowers is 0. Local tobacco spreads rather aggressively, and so on each consecutive day, the tobacco quantity on park tiles grows as follows:

- Tobacco quantity on tiles with tobacco and grass is increased by the sum of tobacco quantity on the neighboring tiles on the previous day.
- Tobacco quantity on tiles with flowers always remains 0.

After some days of growing, the harvest operation will take place on a particular day. The gangsters will choose some of the tiles (of any type) to be harvested and obtain the total tobacco quantity on such tiles on that day. Each tile may be harvested at most once and all tobacco quantity on a chosen tile must be harvested. The gangsters can't harvest everything, because there is a limit on the tobacco quantity that can be somewhat safely smuggled. At the same time, the gangsters do not want to take any unnecessary losses and so they want to harvest exactly the limit amount.

The task is to help the gangsters to plan the operation to the very detail. That is in particular:

1. For the first day, choose flower tiles to be cut, and from these choose tiles to plant tobacco on. At most $2 \cdot 10^5$ flower tiles may be cut, to not raise suspicion among policemen.
2. Choose the number of days to keep tobacco growing. The number of days must be at most 100 to not make this tobacco business blatantly obvious.
3. Choose tiles to be harvested after the given number of days pass. At most 10^4 tiles may be harvested, as any more would take the gangsters too much time.

Will you manage to help the gangsters? Good luck!

Input Specification

The input contains a single line with an integer N ($0 \leq N \leq 10^{18}$), the exact tobacco quantity to be harvested.

Output Specification

On the first line, print the number C ($0 \leq C \leq 2 \cdot 10^5$) of cut flower tiles. For each such tile, output a line in the form of either X Y 1 for a cut tile with integer coordinates (X, Y) and with a planted tobacco, or X Y 0 for a cut tile left as a grass area ($-10^6 \leq X, Y \leq 10^6$).

Afterwards print a line with two space separated numbers H and D ($0 \leq H \leq 10^4$, $0 \leq D \leq 100$), the number of harvested fields and the number of days for tobacco to grow, respectively. Then print H lines with two space separated integers X and Y ($-10^6 \leq X, Y \leq 10^6$), the coordinates of tiles to be harvested after D days.

Sample Input 1

```
1
```

Output for Sample Input 1

```
1
0 0 1
1 0
0 0
```

Sample Input 2

```
13
```

Output for Sample Input 2

```
8
1 0 1
0 1 0
1 1 0
3 1 0
1 2 1
2 2 1
1 3 1
3 3 1
3 2
3 3
1 2
1 0
```

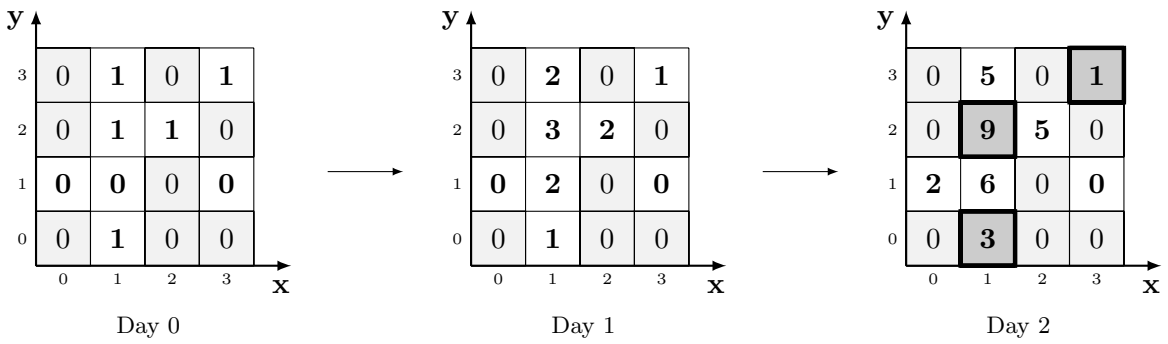


Figure 1: Illustration of Sample Input 2

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Rescue Mission

`mission.c`, `mission.cpp`, `Mission.java`, `mission.py`

The plan of the Criminal Liberating Rough Squad (CLRS) is to attack the train carrying the prisoners to another jail through a desert and to free at least some of the them.

CLRS got exactly 10 trucks which will carry the rescued criminals from the place of the assault to the makeshift airport where the planes are already being refueled before they fly abroad with the criminals on board.

At the assault scene, the CLRS will break into a train coach, neutralize coach guards, free all prisoners in the coach and move to the next coach. The squad will progress from the first attacked coach towards the end of the train, liberating the criminals in one coach after another. CLRS is proud to claim they are going to free and load into the trucks all criminals from any coach they will attack. CLRS moves in the train in only one direction, they never turn back.

Somewhat strangely, when the trucks will be leaving the scene, the number of freed criminals has to be exactly the same in all trucks. It is an old time safety superstition of CLRS and it cannot be broken at any cost at any action of this kind.

There are bad news too. The police will probably be patrolling relatively nearby, thus the scene has to be left as soon as possible after the initial assault. That is, immediately when the superstition rule allows it.

It may also happen, the mission will be impossible to accomplish. For example if CLRS starts the attack on a coach too close to the end of the train.

Now, everything has to be planned carefully. The number of transported criminals in each coach is known to CLRS beforehand. They want to know, for each coach in the train, how many coaches will they have to attack if the assault starts on that particular coach.

Input Specification

The first input line contains integer N ($1 \leq N \leq 10^5$), the number of coaches in the train. The second line contains N values between 0 and 9 (inclusive), the number of transported criminals in each coach. The values are listed in the order from the first coach in the train to the last.

Output Specification

The output consists of sequence of N numbers, k-th value in the sequence is the number of attacked coaches when the assault starts on k-th coach. If the mission cannot be accomplished starting on k-th coach, the corresponding value in the sequence is -1.

Sample Input 1

```
5
0 2 4 6 8
```

Output for Sample Input 1

```
1 4 2 -1 -1
```

Sample Input 2

```
5
5 5 5 0 5
```

Output for Sample Input 2

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

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Offices

`offices.c`, `offices.cpp`, `Offices.java`, `offices.py`

The network of detective offices is growing. New offices are connected to the already existing offices via computer cables. There are two types of cables – optical and quantum – which propagate signal at different speeds. It takes time T_1 and T_2 for a signal to travel between two offices connected by an optical cable and by a quantum cable, respectively. There can be at most one cable between any two offices and it must be of one of the two types.

There are two ranks of detectives, Technical rank and Social rank. Each detective belongs to exactly one rank.

The process of building and installing a new office is the following. When a request for a new office is raised and it complies with the request rules, a new office is built and gradually populated by detectives of both ranks.

The request for a new office comes from two detectives, say A and B, who have different ranks and who also reside in different offices. Suppose the first detective resides in office OA and the second one in office OB. For safety reasons, the offices OA and OB must not share a connection. It is guaranteed by the administration that no request from two offices sharing a connection is possible, as every officer understands the dangers of such a request.

Suppose that ON is the office that will be newly built. Office ON can be connected only to those already existing offices (denote by OE any one of them), to which at least one of OA and OB is already connected. Office ON will be connected to all offices satisfying the conditions and specifications given below.

The members of both ranks have opposing views on the reliability of the cables. A member of the Technical rank thinks that the type of connection between ON and OE should be the same as the type of connection of his current office to OE. A member of the Social rank thinks that the type of connection between ON and OE should not be the same type as the type of connection of his current office to OE. Therefore, the Headquarters have set the rules regarding new cables:

- When OE is connected to only one of the offices OA and OB, the type of the connection between ON and OE is decided by the detective in the respective office connected to OE.
- When OE is connected to both offices OA and OB, and the detectives A and B disagree on the type of the connection between ON and OE, then the connection is not built.
- When OE is connected to both offices OA and OB, and the detectives A and B agree on the type of connection between ON and OE, then the connection is built. However, it should be stressed, it is of the other type than the one which A and B agree on.

Now, there is a need of an application which keeps track of all the offices and the cables connecting them.

The Headquarters are located in the office labeled 0. After building a new office, the headquarters must be informed on the total sum of distances between the Headquarters and all the other offices that can be reached from it by a sequence of connections. The distance between two offices is the shortest time in which the signal can travel via cables from one office to another, supposing it spends no additional time inside any office on its path.

Input Specification

The first line of input contains five integers N, M, R, T_1, T_2 ($1 < N \leq 5; 0 \leq M \leq \binom{N}{2}; 0 \leq R \leq 10^5; 0 \leq T_1, T_2 \leq 100$), N is the number of existing offices, M is the total number of cables between the existing offices, R is the number of new office requests, T_1 and T_2 is the time a signal travels through an optical and a quantum cable, respectively.

The offices are labeled as $0, 1, \cdots, N-1$. Each of the next M lines contains two integers a and b ($0 \leq a \neq b < N$) and a character "`O`" or "`Q`". Integers determine the labels of the connected offices, the character determines the type of the cable which connects them (optical or quantum).

Next, there are R lines, each of which represents one request to build a new office. The i-th request line (request counter i starts from 0) contains two integers a and b ($0 \leq a \neq b < N + i$), the labels of the offices in which resides the requesting pair of detectives. It is assumed that the office occupied by the detective of Technical rank is always listed first. Furthermore, it is guaranteed that no request will be between two offices that share a direct connection.

Output Specification

For each request in the input, print on a separate line the sum of distances from the Headquarters to all other offices that can be reached from the Headquarters by a sequence of connections.

Sample Input 1

```
4 3 1 1 100
0 1 Q
1 2 Q
2 3 Q
3 1
```

Output for Sample Input 1

```
601
```

Sample Input 2

```
5 3 2 1 100
0 1 Q
1 2 O
3 4 O
0 2
5 4
```

Output for Sample Input 2

```
302
806
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Pickpockets

`pickpockets.c`, `pickpockets.cpp`, `Pickpockets.java`, `pickpockets.py`

The police station stands at the very top of the Jewellery Stores lane, a location vitaly important for the pickpocket business in the town. The policemen start their daily patrol at the top end of the lane, progress slowly down and then return back to the top, seldom reaching the bottom end of the lane during the day. There are many regularities in the police habits, and therefore Big Pickpocket Boss (BPB) can make a smart plan for the holidays. The stores on the lane are labeled by successive integers, beginning from 1, from the bottom of the lane to its top. For each day of holidays, BPB can guarantee some number of stores, from the beginning up to a particular label, to be clean from the police unwelcome scrutiny. Teams of pickpockets will do the job for BPB. There are many teams available, each of them can operate on a single store for a number of consecutive days. Not necessarily all teams must be employed.

BPB is a formidable boss, his rules must be obeyed to the letter:

- There will be exactly one team operating in each store on any day the store is clean.
- When a team starts operating in a store they will operate there for some number of consecutive days.
- When a store will not be clean on a particular day, no team will operate in the store on that day.
- No team will operate in two or more stores.
- No team will operate twice or more times during the holidays.
- No team will operate on any day before or after the holidays.

It is known that each team can generate their specific minimum income for BPB during their entire operation. BPB knows he has to maximize his minimum total income. He wants that figure from you, today, by 3 PM and not later. Do not even try to disappoint him.

Input Specification

The first line contains two integers H and T ($1 \leq H \leq 10^5, 1 \leq T \leq 16$), the number of days in the holidays and the number of teams available. The second line contains H integers C_k ($0 \leq C_k \leq 10^5, 1 \leq k \leq H$), the highest labels of a clean store on the k-th day of holidays. Label 0 means there is no clean store on the k-th day of holidays. Each of the next T lines contains two integers D_t and I_t ($1 \leq D_t \leq H, 0 \leq I_t \leq 10^6$, $1 \leq t \leq T$), the duration of the operation of team t in days, and the minimum income the team generates.

Output Specification

Print the maximum value of the minimum total income the teams can generate when appropriately scheduled. Print 0 when the conditions of BPB cannot be met.

Sample Input 1

```
3 4
2 1 2
3 2
1 1
1 2
1 3
```

Output for Sample Input 1

```
7
```

Sample Input 2

```
4 7
2 2 1 1
3 1
1 1
1 4
1 1
2 4
2 2
2 1
```

Output for Sample Input 2

```
11
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Storage Problems

`problems.c`, `problems.cpp`, `Problems.java`, `problems.py`

The gangsters did a very successful robbery of the city's most famous auction house. Now they are safely at their hideout, where they store the stolen items. Luckily, you managed to place a listening device into their hideout. You also have a personal file on each ganger, which contains a recording of their voice. You will listen carefully to what happens next with hope it will help you with the investigation of the robbery.

Each gangster stole exactly one item, the i-th gangster stole the i-th item. Now each gangster is trying to put his item into the common storage, which can hold a total weight of K. The storage is a small room and the gangsters store their items one by one.

When a gangster tries to put an item into the storage but it does not fit, that is the total weight of the items in the storage would exceed K, he gets angry and throws all the items in the storage out. While doing this, he tells the others that "j items are going to trash!", where j is the number of items in the storage at the point he tried to store his item. At this point a fight ensues and no more storing will happen.

As you have a listening device in the gangsters' storage, you will hear how much items the gangster throws out. Also, using your personal files, you can tell apart each of the gangster's voices.

Therefore, it would help your investigation greatly if you could know in advance, for all possible values of j and i, how many different subsets of items could be in the storage at the moment when the i-th gangster throws all the j items out. As the number of subsets can be large, output it modulo 167772161.

Input Specification

The input consists of two lines. The first line contains two integers N and K ($2 \leq N \leq 400$, $1 \leq K \leq 400$), the number of gangsters and the maximum weight that the storage can hold. The second line contains N integers $w_1, w_2, \cdots, w_N$, such that ($1 \leq w_i \leq K$) for each $1 \leq i \leq N$. Here w_i is the weight of item that the i-th gangster stole.

Output Specification

The output consists of N lines, each line containing exactly $N - 1$ integers. The j-th value on the i-th line contains the number of subsets of items containing exactly j items, such that they fit into the storage but the i-th gangter's item can not be added. Each number is modulo 167772161.

Sample Input 1

```
3 3
2 2 1
```

Output for Sample Input 1

```
1 1
1 1
0 0
```

Sample Input 2

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

Output for Sample Input 2

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

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Roof Escape

`roof.c`, `roof.cpp`, `Roof.java`, `roof.py`

Escaping from the police over city roofs is often tricky and the gangsters have to be trained properly. To keep up with current AI trends in criminality, they are developing a general computerized model of escape paths.

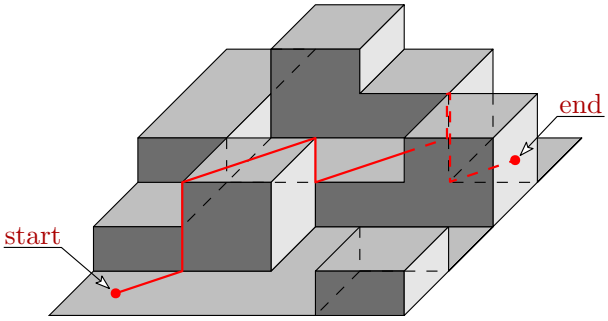
In the model, the city area where the escape happens is modeled as a 3D grid made of rectangular cuboids with square bases forming a 2D grid on flat ground. Each cuboid represents a block of houses. Top face of a cuboid is called a roof. In the model, all distances between adjacent blocks are reduced to 0. The path of escaping gangsters is modeled as a polyline – a sequence of straight horizontal and vertical segments where the end point of one segment is the start point of the next segment. The basic path properties are:

- Each point on the path is on the surface of at least one block.
- No part of the path is in the interior of any block.
- The height of any point on the path is bigger than or equal to the lowest height of roofs of all blocks to which surface the point belongs.
- The path starts and ends in the center of a block roof.
- The sum over the lengths of horizontal segments of the path is minimum possible.

It may happen that two consecutive segments on the path share common points. This stems from the fact that the path models a real behavior of a person moving over physical obstacles. Thus an additional path rule also holds:

- Let P be a point on the path. If there is a point Q directly above P, and Q belongs to at least two blocks, then the point Q is on the path.

The total length of the escape path should be carefully calculated in the model.



Input Specification

The first line of the input contains six positive integers W, H, S_x, S_y, E_x, E_y ($1 \leq W \cdot H \leq 10^5$, $1 \leq S_x, E_x \leq W$, $1 \leq S_y, E_y \leq H$). W and H are even integers representing the dimensions of the grid base in meters, integers S_x, S_y denote starting coordinates of the escape path and E_x, E_y denote coordinates of the end.

Each of the next $H/2$ lines contains $W/2$ integers, the i-th integer on j-th line is the height of the corresponding block $T_{i,j}$ in meters ($0 \leq T_{i,j} \leq 10^3$).

Each grid block base is a square with dimensions of 2×2 meters in the model.

Output Specification

Print the length of the escape path. The difference between the printed length and the exact length must be less than 10^{-4}.

Sample Input 1	Output for Sample Input 1
8 8 1 7 7 1 2 3 2 0 2 1 1 2 1 2 0 0 0 0 0 1	14.485281374238

The sample input with its solution is shown on the picture above.

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter

Central Europe Regional Contest 2020

Screamers

`screamers.c`, `screamers.cpp`, `Screamers.java`, `screamers.py`

The police is preparing a huge sting in which they hope to jail most of the prominent criminal figures in the city. The flow of information on the side of the police has to be as tight as possible to prevent any leaks. Each detective officer (DO) taking part in the sting is under a strict regulation.

The information shared among DOs is in the form of so-called drops. A drop is always spoken, it must not be recorded on any medium, electronics, paper, etc. Any DO can share a drop only with selected DOs with which he shares a bidirectional connection. Each DO is obliged to pass the drop, as soon as possible and without any change, to all his buddies with which he shares a connection, except for the DO from which he received the drop.

The Chieft Inspector (CI) has to choose which pairs of DOs will share a connection. This final set of connections is called the final group. In this final group (FG) an additional FG-rule holds: A situation when a drop returns to a DO who passed it to his buddies some time ago must not happen. It would mean there are too many unnecessary connections in the FG network.

There is a stack of folders, each folder describing one connection between a particular pair of DOs. The selection of FG is done in two steps. First, CI chooses two integer values S and T, which may be sometimes the same, and removes from the stack all folders above the S-th folder and all folders below the T-th folder.

Next, with the reduced folders, CI repeats the operation. He chooses two integer values U and V, which may be sometimes the same, and removes from the reduced stack all folders above the U-th folder and all folders below the V-th folder.

CI wants to use all the connections in the remaining folders in the FG. However, it is not guaranteed that connections can form FG, due to the additional FG-rule. CI tends to forget this rule quite often.

One cannot change the professional habits of CI. His assistant tries to address the issue diplomatically by employing a programmer who would gradually computerize the process. His first task is to compute the number of different FGs that may be selected by CI after he has chosen the first two values S and T. This computation must be efficient for many different values of S and T.

Input Specification

The first input line contains two numbers, N and M ($1 \leq N, M \leq 10^5$), the number of DOs and the number of folders in the CI's stack respectively. The DOs are identified by integers $1 \ldots N$. Next, there are M lines, each represents one folder and it contains two integers A and B ($1 \leq A < B \leq N$), pair of DOs whose connection is described in the folder. The order of lines corresponds to the order of folders in the stack from top to bottom.

The next line contains one number Q ($1 \leq Q \leq 10^5$), the number of queries. Next, there are Q lines, each represents one query and it contains two positive integers S and T ($1 \leq S \leq T \leq M$), the numbers chosen by CI in the first step of FG selection.

Output Specification

For each of the Q query input lines print the number of different FGs which can be formed in the second step of the selection process.

Sample Input 1

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

Output for Sample Input 1

```
1
5
6
13
```